НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 004.89

«До захисту допущено»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___»_____ 2021 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-науковою програмою
«Інженерія програмного забезпечення комп'ютерних
та інформаційно-пошукових систем»

зі спеціальності 121 Інженерія програмного забезпечення
на тему: «Адаптивні методи навчання нейромереж за наявності обмежених обчислювальних ресурсів»

Виконав:
студент ІІ курсу, групи КП-91 мн
Іващенко Михайло Вікторович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н.,
Люшенко Леся Анатоліївна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри ММСА ІПСА, к.ф.-м.н., доцент,
Шубенкова Ірина Анатоліївна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра прикладної математики

Рівень вищої освіти – другий (магістерський)

Спеціальність (спеціалізація) – 121 «Інженерія програмного забезпечення»

Освітньо-наукова програма «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Науковий керівник

_____ Іван ДИЧКА

«___»_____2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Іващенку Михайлу Вікторовичу

1. Тема дисертації «Адаптивні методи навчання нейромереж за наявності обмежених обчислювальних ресурсів», науковий керівник дисертації Люшенко Леся Анатоліївна, к.т.н., доцент, затверджені наказом по університету від «26» травня 2021 р. №899-С.

2. Термін подання студентом дисертації «18» травня 2021 р.

3. Об'єкт дослідження: процес адаптивного навчання нейромереж.

4. Предмет дослідження: методи адаптивного навчання нейронних мереж для вирішення задачі класифікації в системах обмежених обчислювальних ресурсів.

5. Перелік завдань, які потрібно розробити:
   - Дослідити існуючі методи навчання та верифікації нейромереж;
   - виявити недоліки використання даних методів при роботі із системами обчислювальні ресурси яких є обмеженими;
   - розробити методи адаптивного навчання нейромереж та критерію оцінки навченості нейромереж для зменшення витрат ресурсів та часу, що використовуються в рамках циклу навчання;
   - розробити синтетичну модель генерації даних для тестування сформованих наукових гіпотез;
   - розробити програмне забезпечення для навчання моделей нейронних мереж та генерації синтетичних даних;
   - реалізувати і протестувати сформульовані методи та критерій на моделі синтетичних даних, використовуючи розроблене програмне забезпечення.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
   - теоретичні аспекти побудови математичної моделі методів адаптивного навчання нейронних мереж на основі композиційної архітектури;

- демонстраційні приклади даних навчання;
- таблиці із порівняльними результатами;
- схема роботи алгоритмів адаптивного навчання нейронних мереж;
- схема роботи алгоритмів верифікації нейронних мереж.

7. Орієнтовний перелік публікацій:
   - Стаття “Artificial Neural Network Training Criterion Formulation Using Error Continuous Domain”
   - Тези доповіді “Критерій оцінки навченості нейронної мережі на основі аналізу похибки передбачень ”

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент кафедри ПЗКС		

9. Дата видачі завдання «11» жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	17.12.2019	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	04.03.2020	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	16.05.2020	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення; підготовка матеріалів доповіді на конференції ПМК-2020	14.10.2020	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	15.12.2020	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	20.02.2021	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу; робота над розділом з охорони праці	16.04.2021	
8.	Оформлення текстової і графічної частини магістерської дисертації	06.05.2021	

Студент Михайло ІВАЩЕНКО

Науковий керівник Леся ЛЮШЕНКО

РЕФЕРАТ

Актуальність теми. Одним з основних аспектів, на якому зосереджуються розробники штучних нейронних мереж, є пошук оптимальної структури (та гіперпараметрів) для вирішення поставленої проблеми. Цей підхід корисний за наявності значної кількості обчислювальних ресурсів (центр обробки даних, набір відеокарт, тощо). Однак, коли обчислювальні ресурси обмежені (або у конкретному середовищі – мікроконтролери/мікрокомп'ютери, або через їх відсутність), необхідно використовувати адаптивні методи навчання для зменшення кількості обчислювальних операцій. Програмне забезпечення, що навчає нейронні мережі шляхом використання адаптивних методів, будує їх на основі композиційних архітектур, використовуючи вхідні дані у вигляді текстових файлів. Кожний текстовий файл відповідає навчальному набору даних, який було згенеровано у відповідності до спроектованої та реалізованої моделі генерації синтетичних даних.

Об'єктом дослідження є методи адаптивного навчання нейронних мереж за наявності обмежених обчислювальних ресурсів для вирішення задачі класифікації.

Предметом дослідження є програмне забезпечення для адаптивного навчання нейронних мереж за наявності обмежених обчислювальних ресурсів.

Мета роботи: використовуючи програмні засоби, створити та дослідити адаптивні методи навчання нейромереж за наявності обмежених обчислювальних ресурсів на основі композиційної архітектури та визначення критерію оцінки навченості нейронної мережі.

Методи дослідження: в роботі використовуються методи теоретичного дослідження: аналіз та синтез. Також застосовувалися емпіричні методи: експеримент та порівняння.

Наукова новизна роботи полягає у тому, що програмне забезпечення, яке використовує розробленні методи адаптивного навчання нейронних мереж на основі композиційної архітектури та критерію навченості нейронних мереж, підвищує точність класифікації на 10-15%.

Практична цінність отриманих результатів роботи полягає в тому, що запропоновані методи надають можливість використовувати підмережі в рамках інших задач, дозволяючи розробляти програмні інструменти для нейронних мереж, які підвищують точність класифікації при збереженні кількості обчислювальних операцій. Запропонований критерій оцінки навчанності дозволяє ідентифікувати піднабори прикладів, точність класифікації яких нейронною мережею є нижчою за допустиму.

Апробація роботи. Основні положення і результати роботи були представлені на XII науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2020 та опубліковані у збірнику тез доповідей.

За результатами роботи була подана наукова стаття на тему «Artificial Neural Network Training Criterion Formulation Using Error Continuous Domain» до наукового міжнародного журналу «International Journal of Modern Education and Computer Science (IJMECS)», яка буде опублікована протягом травня 2021 р. Очікується підтвердження на проведення відповідної конференції у червні 2021.

Структура та обсяг роботи. Магістерська дисертація складається з вступу, чотирьох розділів, висновків та додатків.

У вступі надано загальну характеристику роботи, виконано оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень.

У першому розділі наведено огляд існуючих підходів до навчання нейронних мереж та їх верифікації в рамках вирішення задачі класифікації та описані їх недоліки.

У другому розділі запропоновано адаптивні методи навчання нейронних мереж на основі композиційних архітектур та критерій оцінки наченості нейронної мережі, що аналізує результати класифікації як контінуальну множину.

У третьому розділі сформовані основні вимоги до програмного комплексу, що включає в себе реалізацію та тестування запропонованих методів і критерію, наведено опис програмної реалізації: список модулів, їх опис та зв’язок, список технологій, фрагменти коду, тощо.

У четвертому розділі визачено критерії оцінки ефективності, які було застосовано до розробленого методу; наведена інформація про дані, що використовувались при аналізі ефективності; проведений аналіз ефективності роботи запропонованих методів навчання нейронних мереж у порівнянні із методом навчання одиночної мережі; надано аналіз ефективності запропонованого критерію навченості нейронних мереж.

У висновках проаналізовано отримані результати роботи.

Ключові слова: нейронні мережі, задача класифікації, навчання нейронних мереж, методи адаптивного навчання нейронних мереж, методи верифікації нейронних мереж, інструменти програмної реалізації, програмне забезпечення, критерій оцінки навченості нейронних мереж, композиційна архітектура.

ABSTRACT

Actuality. One of the main aspects on which the developers of artificial neural networks focus is searching for the optimal architecture (and hyperparameters) to solve the classification problem. This approach is useful in the presence of a significant amount of computing resources (data center, a set of video cards). However, when resources are limited (either in a specific environment – microcontrollers/microcomputers, or due to their absence), it is necessary to use adaptive learning methods to reduce the number of computational operations. The developed software that trains neural networks using adaptive methods builds them on composite architectures using input as text files. Each text file corresponds to a training data set generated according to the designed and implemented model of synthetic data generation.

Object of research is a process of adaptive learning of neural networks in the system with limited number of resources.

Subjects of research is a software for adaptive learning of neural networks in the presence of limited computing resources.

Goal of the work is to implement and research adaptive methods of neural network learning in the presence of limited computing resources on the basis of composite architecture and definition of a criterion of an estimation of a neural network learning.

Methods of research include the methods of theoretical research that are used in the thesis: analysis and synthesis. The following empirical methods were also used: experiment and comparison.

Scientific novelty of the work is that the software, which uses the development of methods of neural network adaptive learning based on the compositional architecture and the criterion of neural networks learning, increases the accuracy of classification by 10-15%.

Practical value of the obtained results of the work is that the proposed methods will allow to develop software tools for neural networks, which

increase the accuracy of classification while maintaining the number of computational operations and provide the opportunity to use subnets in other tasks. The proposed criterion for assessing learning allows to identify subsets of examples whose accuracy of classification provided the neural network is lower than acceptable.

Approbation. The main provisions and results of the work were presented at the XII scientific conference of undergraduates and graduate students "Applied Mathematics and Computing" PMK-2020 and published in the collection of abstracts. As a result, a scientific article on "Artificial Neural Network Training Criterion Formulation Using Error Continuous Domain" was submitted to the International Journal of Modern Education and Computer Science (IJMECS), which will be published in May 2021. Confirmation is expected on holding a conference in June 2021.

Structure and content of the thesis. Master's thesis consists of an introduction, four chapters, conclusions, and appendices.

The introduction provides a general description of the work, evaluates the current state of the problem, substantiates the relevance of the research direction, formulates the purpose and objectives of the study.

The first section provides an overview of existing approaches to the learning of neural networks and their verification in solving the classification problem and describes their shortcomings.

The second section proposes adaptive methods of neural network learning based on composite architectures and the criterion for estimating the completeness of the neural network, which analyzes the results of classification as a continuous set.

The third section forms the basic requirements for the software package, including implementing and testing the proposed methods and criteria, a description of the software implementation: a list of modules, their description and connection, a list of technologies, code fragments, etc.

The fourth section provides criteria for evaluating the effects that were applied to the developed method; provides information on the data used in the performance analysis; the analysis of efficiency of work of the offered methods of training of neural networks in comparison with a method of training of a single network is carried out; the analysis of efficiency of the offered criterion of training of neural networks is given.

The conclusion contains a brief overview of the results obtained in the thesis.

Keywords: neural networks, classification problem, training of neural networks, methods of adaptive training of neural networks, verification of neural networks, a criterion for assessing the learning of neural networks, compositional architecture.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Безпечна властивість – набір лінійних обмежень, який не має перетинатись із вихідним набором нейронної мережі.

Вентильний рекурентний вузол (GRU) – механізм затвора в рекурентних нейронних мережах.

Глибоке навчання (також відоме як глибоке структуроване навчання) є частиною більш широкого сімейства методів машинного навчання, заснованих на штучних нейронних мережах з репрезентативним навчанням.

Довга короткочасна пам'ять (LSTM) – це штучна архітектура рекурентних нейронних мереж (RNN), що використовується в галузі глибокого навчання.

Доступний набір шару – результат обчислювальних операцій, застосованих шаром до відповідних вхідних даних.

Дропаут – метод регуляризації для зменшення перенавчання в штучних нейронних мережах шляхом запобігання складним ко-адаптаціям даних навчання.

Індуктивне упередження алгоритму – це набір припущень, які використовує учень для прогнозування результатів з урахуванням введених даних, з якими він ще не стикався.

Лінійне програмування – область методів пошуку оптимальних результатів (таких як максимальний прибуток або найнижча вартість) в математичній моделі із лінійними обмеженнями.

Логічне вирішення конфліктів – розв’язання задач за допомогою булевої логіки.

Логічне розбиття – розбиття задачі на підзадачі, використовуючи правила логічної мови.

Логічне слідування – це фундаментальне поняття в логіці, яке описує взаємозв'язок між твердженнями, які відповідають дійсності, коли одне твердження логічно слідує з одного або декількох тверджень.

Локалізація – процес визначення місцеположення об’єкта на сцені.

М’який обмін параметрами – модель прагне вивчити спільне або “подібне” приховане представлення для різних завдань. Для того, щоб встановити подібність між завданнями, модель навчається одночасно для всіх завдань і з певними обмеженнями або регуляризацією взаємозв'язку між пов'язаними параметрами.

Маппінг – процес побудови топологічної карти невідомої середи в процесі реального часу.

Маркерований – застосовується по відношенню до навчального приклада. Ідентифікує, що для кожного з об’єктів, який присутній у прикладі надана інформація про відповідний клас.

Поліедрон (багатогранник) – це тривимірна фігура з плоскими багатокутними гранями, прямими ребрами та гострими кутами або вершинами.

Регуляризація – в математиці, статистиці, фінансах, інформатиці, особливо в машинному навчанні та зворотних проблемах, – це процес додавання інформації з метою вирішення неправильно поставленої проблеми або запобігання перенавчанню.

«Розумний» алгоритм – алгоритм, що володіє сукупністю припущень, які він використовує для прогнозування результатів на основі вихідних даних, з якими він не зустрічався.

ВСТУП

Тенденцією розвитку IT технологій є впровадження нейронних мереж в сучасні програмні комплекси [1]. Одними з популярних задач, які вирішують за допомогою використання нейронних мереж є задачі класифікації [2] та кластеризації [3]. Задача класифікації полягає у побудові бієктивного відношення «приклад – клас». Рішення цієї задачі може бути реалізовано за допомогою використання наступних видів нейронних мереж: персептрон [4], згорткові нейронні мережі [5], залишкові нейронні мережі [6], рекурентні нейронні мережі [7].

Одним з основних аспектів, на якому зосереджуються розробники штучних нейронних мереж, є пошук оптимальної структури (та гіперпараметрів) для вирішення поставленої проблеми. Цей підхід корисний за наявності значної кількості обчислювальних ресурсів (центр обробки даних, набір відеокарт, тощо). Однак, коли ресурси обмежені (або у конкретному середовищі – мікроконтролери/мікрокомп’ютери, або через їх відсутність), такий цикл навчання значно втрачає продуктивність або стає абсолютно неможливим у використанні.

Для вирішення цієї проблеми можна скористатися підходом композиції нейронної мережі. Дослідження показали, що зменшення кількості нейронів може підвищити точність розпізнавання [4, 5]. Окрім цього, композиційна архітектура дозволяє методу навчання та отриманій нейронній мережі набори властивості адаптивності. Дана властивість полягає у можливості конфігурації списку підазадач і, як результат, підмереж, які розв’язують ці підзадачі. В результаті, стає можливим розв’язання різних піднаборів задач, так як архітектура нейронної мережі дозволяє додавати та вилучати підмережі в залежності від формулювання задачі (підмережі мають бути незалежні одна від одної).

Другим важливим аспектом циклу розробки нейронної мережі є стадія її тестування або стадія верифікації. Дана стадія лягла в основу проблеми

верифікації. Рішення проблеми верифікації нейронних мереж [8] отримали стрімкий розвиток, з'явившись в рамках необхідності побудови критерію навченості нейронної мережі. В даний момент використовується серія підходів для вирішення задачі верифікації [9]. Тим не менш, в загальному вигляді вона не вирішена, а підходи, які використовуються не включають аналіз специфіки розподілу даних в рамках вхідного та вихідного наборів даних.

Дана дисертація пропонує нові метод адаптивного навчання нейронних мереж на основі композиційної архітектури та критерій оцінки навченості нейронної мережі. Метою методу адаптивного навчання є зменшення кількості обчислювальних операцій в рамках процесу навчання шляхом надання підмережам статусу взаємозамінності. Метою запропонованого критерію навченості є розгляд похибки класифікації як контінуальної множини із подальшою ідентифікацією піднаборів прикладів, точність класифікації яких є нижчою за допустиму.

Дисертація надає опис програмного комплексу, який було розроблено для тестування сформульованих гіпотез. Розроблена реалізація складається з п'яти модулів, які включають передоброку навчальних наборів, побудову моделей, тестування та візуалізацію. Приймаючи на вхід відповідні, програмний комплекс забезпечує компіляцію та навчання нейронних мереж у відповідності з описаними методами та візуалізує результати. Даний комплекс було розроблено мовою Python із використанням таких бібліотек, як Tensorflow, Keras, Scikit-Learn, тощо.

1. ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ ДО НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ ТА ЇХ ВЕРИФІКАЦІЇ

Не дивлячись на те, що Google, Microsoft, Amazon та інші компанії відкривають свої онлайн-ресурси для навчання нейронних мереж, цей процес потребує удосконалення та розвитку. Більше того, проблемою залишається організація навчання на платформах, які володіють обмеженим об’ємом ресурсів (наприклад, мікрокомп’ютери). Дані платформи, зазвичай, будуються на основі мікроконтролерів та мікрокомп’ютерів, що суттєво підвищує рівень їх портативності. До таких систем відносяться:

- «Розумні» мультисенсорні системи, які будуються на основі архітектур із великою кількістю датчиків, що оброблюють чисельні джерела інформації. Дана модель дозволяє приймати ті чи інші рішення, обумовлені організацією середовища, в якому функціонує система («розумний» дім, системи охорони, мульсимедійні системи, тощо).
- Автономні «розумні» агенти, які представляють собою роботизовані пристрої малих або середніх габаритів, що мають навчатись самостійно, віддалено від основної системи. Система, розміщена всередині агента, реалізовує як алгоритм розпізнавання, так і алгоритм для самостійного навчання. Прикладами таких рішень є «розумні» безпілотники, автопілоти, роботизовані міні-кари тощо.

Існуючі підходи не є оптимальними до навчання зазначених вище нейронних мереж. Як правило, такі мережі не мають централізованого обчислювального центру, що дозволяє виконувати достатню кількість операцій за одиницю часу.

Окрім того, процес розробки таких систем викликає певні труднощі, а саме:

- Розгортання середи розробки. Зазвичай, подібні системи реалізовуються на основі пристроїв, що мають специфічні конфігурації та вимоги до встановлення бібліотек.
- Навчання в робочому середовищі. Для того, щоб система могла успішно навчатись в режимі реального часу, необхідно надати їй певний об'єм пам’яті та обчислювальні ресурси. Причиною цього є обчислювально витратні операції при роботі із нейронними мережами. Дані ресурси розподіляються як на безпосереднє виконання обчислень, так і на підтримку сторонніх функцій, які реалізовуються обраною бібліотекою.

Таким чином актуальним є створення адаптивного методу навчання, який дозволить з одного боку оптимізувати процес навчання нейронної мережі, а з іншого удосконалити процес розробки.

Іншою проблемою, з якої зтикаються при роботі із нейронними мережами, є відсутність повноцінних методів оцінки якості роботи мереж. До останнього часу в якості даних методів використовувались емпіричні оцінки (точність, *F*-міра, тощо). За декілька останніх років з’явились точніші та складніші методи, які дозволяють оцінити якість навчання мережі – методи верифікації. Як емпіричні оцінки, так і методи верифікації надають відповідь на питання чи можливо використовувати дану нейромережу для вирішення відповідної задачі. Основним недоліком обох підходів є неспроможність надати детальний опис та пояснення, чому відповідно нейронна мережа підходить чи не підходить. Зазвичай, даний недолік стає критичним, коли нейромережа визначена як непридатна, що означає необхідність перенавчання. Виконуючи кожного разу перенавчання мережі з нуля, розробники витрачають велику кількість часу та ресурсів. Саме тому в рамках дослідження було закладено розробку критерію оцінки якості класифікації, який би надавав деталізацію

отриманого нейромережею статусу (придатна чи непридатна до використання).

Розглянемо детально основні підходи до навчання та верифікації нейронних мереж.

1.1. Характеристика загальних підходів до навчання нейронних мереж

В залежності від специфіки поставлених задач, використовуються різні архітектури нейронних мереж, а саме:

- Повнозв’язні мережі прямого розповсюдження – один із початкових концептів штучних нейронних мереж. Використовуються при обробці статистичних даних, зазвичай, моделюючи рішення для регресійних (як лінійних, так і нелінійних) проблем.
- Згорткові нейронні мережі – використовуються в рішеннях, пов’язаних із обробкою зображень. Підтримують багатокласове розпізнавання, локалізацію та маппінг, тощо.
- Рекурентні нейронні мережі – використовуються при обробці контінуальних даних. Зазвичай, це відноситься до звуку та тексту. Зв'язки між вузлами мережі створюють орієнтований у часі граф, що дозволяє їй відображати динамічну поведінку в часі.
- Генеративно-змагальні мережі – використовують методи машинного навчання, які засновані на поєднанні двох нейронних мереж, одна з яких мережа G генерує приклади – генеративна мережею, а друга мережа D намагається відрізнити правильні приклади від неправильних – дискримінаційна мережа. Оскільки мережі G і D мають протилежні цілі – створювати приклади та відкидати приклади – між ними виникає антагоністична гра.

Не дивлячись на присутність великої кількості різних архітектур нейромереж та відповідних модифікацій, процес навчання на абстрактному рівні залишається незмінним. Процес навчання нейронних мереж може

суттєво відрізнятись в залежності від задач, які необхідно розв'язати. Окрім архітектури на процес розробки впливає розмічування даних, вибір гіперпараметрів нейронної мережі, тощо. Не дивлячись на це, в кожному процесі навчання можливо виділити наступні стадії розробки:

- Визначення типу навчальних прикладів. Перш ніж робити щось інше, користувач повинен вирішити, який тип даних використовуватиметься для формування навчального набору. Наприклад, у випадку аналізу рукописного вводу це може бути один рукописний символ, ціле рукописне слово або цілий рядок слів рукописного вводу.
- Формулювання навчального набору. Навчальний набір повинен відповідати обраній проблемі, будучи побудованим таким чином, щоб покращувати процес навчання нейромережі. Зазвичай, даний набір формується шляхом збору вимірювань (статистичних даних, результатів експериментів, синтезу даних, тощо) з певних ресурсів. Окрім цього, дані можуть надаватись експертами з відповідної галузі.
- Формулювання ознак навчальної (оптимізаційної) функції. Точність навчальної функції залежить від того, яким чином представлений вихідний об'єкт (інакше – приклад). Як правило, вихідний об'єкт перетворюється на вектор ознак, який містить ряд ознак, що описують об'єкт. Кількість ознак не повинна бути занадто великою, але повинна містити достатньо інформації для точного прогнозування результат.
- Вибір алгоритму навчання (векторні машини, градієнтний спуск, тощо).
- Завершення побудови моделі. Алгоритм навчання запускається на відповідному навчальному наборі. Також виконується коригування значень гіперпараметрів шляхом оптимізації продуктивності підмножини (що називається набором перевірки) навчального

набору або за допомогою перехресної перевірки (також крос-валідація).

- Оцінка точності роботи навчальної функції. Після налаштування гіперпараметрів, результативність отриманої функції слід вимірювати на тестовому наборі, який формується окремо від навчального набору.

Пошук оптимальної структури та гіперпараметрів, а також етап навчання займають найбільшу кількість часу із всього циклу побудови моделі. Причиною цього є необхідність перенавчати мережу при кожній зміні, що відбулась в моделі.

В зв'язку з цим розглянемо існуючі методи навчання нейронних мереж.

Навчання із вчителем: маємо множину об’єктів (прикладів) та множину класів. Завданням моделі є формування бієктивного відношення «приклад-клас». Для моделі заздалегідь є відомим навчальний набір (множина пар «об’єкт-клас»). На основі даного набору модель визначає функцію, що може бути використана для обробки прикладів, які не належать до нього. Модель нейромережі надає відповідь із відповідною вірогідністю, яку оцінює «вчитель». Алгоритми навчання із вчителем полягають у співставленні відповідних класів із наданими передбаченнями. При цьому, існують складніші алгоритми, в рамках яких вчитель виконує додаткові маніпуляції із даними. Даний підхід вимагає, щоб алгоритм навчання узагальнював дані навчального набору до невидимих ситуацій «розумним» способом.

Навчання без вчителя включає в себе алгоритми, які навчаються розпізнавати закономірності на основі немаркерованих даних. Підхід полягає в тому, що завдяки імітації програма змушена побудувати компактне внутрішнє представлення наданого їй середовища. На відміну від навчання із вчителем, де даним в навчальному наборі ставиться у відповідність множина класів, навчання без вчителя реалізовує

самоорганізацію, яка фіксує закономірності як нейрональні попередні вилучення або щільність ймовірності [10]. На основі навчання без вчителя було сформовано інші популярні підходи: підсилене навчання, де машині в якості орієнтиру дається лише числова метрика продуктивності, та змішане навчання із вчителем, де лише частина даних вхідного набору є “маркерованою”.

Всі алгоритми (включаючи розроблений), розглянуті в даній дисертації, підпадають під категорію навчання із вчителем. Причина вибору даної категорії методів є вища ступінь детермінованості навчання та легші способи оцінки результатів.

1.2. Характеристика загальних підходів до верифікації нейромереж

Верифікація нейронних мереж як окрема область досліджень виникла через неможливість передбачити їх поведінку. Кожну нейронну мережу можна розглядати як чорний ящик, що створює серйозну перешкоду для їх використання в критично важливих системах безпеки. Нещодавні дослідження (наприклад, [11]) показали, що моделі нейронних мереж з високою точністю передбачення тестових наборів даних, можуть бути уразливі до конкурентних атак (невеликі вхідні обурення, що ведуть до помилкового поведінки). Разом з підвищенням точності класифікації при використанні цих моделей зростає помилка активації окремих нейронів в глибоких шарах нейромережі [12]. Цей факт означає, що, надаючи відповідні приклади нейронної мережі, активація певних нейронів в глибоких шарах мережі може призвести до значних помилок в остаточних розрахунках.

Спроби створення рішень для проблеми верифікації нейромереж [13] призвели до створення критеріїв навченості. В даний час для отримання часткового рішення проблеми верифікації використовується ряд підходів [14]. Тим не менш, в цілому дана проблема не вирішена. До

останнього часу процес аналізу якості ґрунтувався на емпіричних підходах, результат роботи яких не є надійним. В останні декілька років з'явились більш потужні методи верифікації:

- Satisfiability Modulo Theory. Даний метод полягає в тому, що набір вхідних даних (у статті – атомів) може бути вбудований у логічну структуру. Подальший алгоритм виконує логічне розповсюдження, логічне розбиття та логічне вирішення конфліктів (Boolean propagation, case-splitting, and Boolean conflict resolution). Після цього, алгоритм кодує ReLU-операції, використовуючи диз'юнкції.
- Алгоритм адаптивного уточнення та змагального пошуку – вирішує задачу лінійного програмування із вхідними обмеженнями x_l, x_u, та вихідними обмеженнями φ_y. Якщо рішення визначає, що проблема не задовольняє наданим обмеженням (множина рішень не перетинає множину, утворену обмеженнями), отримана модель є «безпечною» для використання у даній задачі.
- Метод абстрактного домену та похідні від нього – вихідний набір представляється у вигляді поліедрону, проходження через кожний шар нейронної мережі відповідає Афінному перетворенню ($W_{i+1,i} \times R_{Li} + b_{i+1}$, де $W_{i+1,i}$ – матриця вагових коефіцієнтів між шарами L_{i+1} та L_i, R_{Li} – результат обчислень попереднього шару, представлений апроксимацією об'єднання поліедронів, b_{i+1} – вектор біасу шару L_{i+1}), переданому у ReLU-операцію: $\mathrm{ReLU}(W_{i+1,i} \times R_{Li} + b_{i+1})$. Результат обчислень останнього шару відповідає вихідному набору даних. Якщо отриманий набір перетинається із попередньо визначеною безпечною властивістю, мережа визнається непридатною до використання.

Сучасні підходи надають відповідь, чи є нейромережа придатною або непридатною для вирішення даної задачі, але не включають аналіз специфіки розподілу даних у вхідних і вихідних наборах. Даний факт

також стає причиною збільшення кількості обчислювальних ресурсів, які витрачаються на перенавчання мережі.

Відповідно, не дивлячись на те, що нейронні мережі є популярним та ефективним інструментом вирішення задач, існує лише невелика кількість методів їх перевірки. Емпіричні метрики не дозволяють виконати повноцінну оцінку на контінуальній множині. Сучасні методи верифікації хоч і вирішують дану проблему, але не надають характеристики того, яким саме чином (на яких прикладах) помиляється мережа.

Одним з завдань дослідження стало формулювання критерія оцінки, який міг би використовуватись для порівняння навченості нейромереж. Даний критерій має оцінювати контінуальну множину, як це роблять сучасні методи верифікації та надавати оцінку навченості мережі.

1.3. Аналіз доцільності використання існуючих способів навчання та верифікації нейронних мереж в рамках систем із обмеженим об’ємом ресурсів

Описані способи навчання та верифікації нейромереж втрачають в ефективності при реалізації в системах із обмеженими обчислювальними. Причиною цього є використання класичного циклу розробки, який вимагає повного перенавчання мережі при отриманні незадовільного результату та наступні недоліки способів навчання:

- Відсутність адаптивної властивості у навчених нейронних мережах через те, що отримані мережі можуть використовуватись виключно для вирішення даної задачі.
- Відсутність повної відповіді на питання придатності використання нейронної мережі. Методи верифікації не надають характеристичну оцінку того, як необхідно змінити структуру мережі або навчальний набір даних, щоб покращити її роботу. Це призводить до повного перенавчання у випадку негативної оцінки.

Повне перенавчання вимагає витрати чисельної кількості ресурсів та часу. Системи, в яких нейронна мережа окремо навчається на апаратному забезпеченні із великою кількістю обчислювальних ресурсів, після чого розміщується в агенті, несуть часові витрати. В системах, де агенту із обмеженим ресурсом пам’яті необхідно навчатись в режимі реального часу, підхід повного перенавчання неможливий для використання. В даній ситуації реалізовані методи мають володіти адаптивною характеристикою (для кожної підзадачі має бути виділена окрема, менша підмережа). Також необхідно, щоб використовуваний метод верифікації надавав оціночну характеристику того, які модифікацій необхідно застосувати до нейронної мережі або навчального набору даних з метою покращення роботи агенту.

1.4. Висновки

1.4.1. *Оптимізований метод навчання*

Глибинні нейронні мережі стали одним з найпопулярніших інструментів розробки автоматизованих рішень. Дані моделі виявились особливо корисними при вирішенні задач класифікації. Тим не менш, сучасні підходи до формування структури мережі та навчання вимагають суттєвих часових витрат. Перенавчання мереж, за необхідності, відбувається на всіх прикладах навчального набору даних. Дані приклади включають і ті класи, які мережа успішно класифікує (що призводить до виконання зайвої роботи).

Необхідно сформувати методи навчання, які могли б оптимізувати процес навчання з точки зору кількості необхідних для виконання обчислювальних операцій. Окрім цього, метод має володіти властивістю адаптивності, щоб результати навчання (включно із отриманими моделями), можливо було використовувати для інших задач.

1.4.2. *Формальний критерій навченості нейромережі*

Сучасні підходи до верифікації нейронних мереж не дають повної відповіді на питання придатності використання нейронної мережі для вирішення поставленої задачі класифікації. Більше того, дані методи не надають відповідного аналізу, наскільки та відповідно до яких метрик нейронна мережа не відповідає сформульованим вимогам.

З іншого боку, емпіричні оцінки точності класифікації нейронних мереж не представляють собою детерміновані алгоритми, які надавали б відповідь для контінуальної множини прикладів. Механізм розрахунку точності класифікації на основі поданих прикладів представляє собою дискретну множину і не може використовуватись для оцінки вхідного набору.

Відповідно, необхідно сформулювати критерій, який би надав характеристичну оцінку навченості мережі на контінуальному наборі даних.

2. АДАПТИВНИЙ МЕТОД НАВЧАННЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ НА ОСНОВІ КОМПОЗИЦІЙНОЇ АРХІТЕКТУРИ

Адаптивний метод навчання нейронних мереж відносяться до методів навчання штучних нейронних мереж на основі композиційної архітектури. Окрім того, при наявності обмежених обчислювальних ресурсів вони використовуються з метою зменшення кількості обчислювальних операцій та збереження оперативної пам’яті. Властивість адаптивності полягає у можливості конфігурації списку підазадач і, як результат, підмереж, які розв’язують ці підзадачі. В результаті, стає можливим розв’язання різних піднаборів задач, так як архітектура нейронної мережі дозволяє додавати та вилучати підмережі в залежності від формулювання задачі (підмережі мають бути незалежні одна від одної).

Використовуючи подібний підхід, стає можливим збереження обчислювальних ресурсів за рахунок вилучення синаптичних зв’язків, які були б присутні у загальній мережі. Дане збереження дозволяє виконувати навчання окремих підмереж в режимі реального часу, що суттєво підвищує ефективність загального процесу навчання.

2.1. Задача класифікації

Під задачею класифікації в рамках даної проблеми будемо розуміти наступне – маємо кінцеву множину об’єктів, кожен з яких ідентифікується певним класом з відомої множини класів, сформульованої у проблемі. Дану множину об’єктів будемо називати навчальним набором даних. Необхідно побудувати алгоритм, здатний поставити у відповідність певний клас кожному з об’єктів множини прикладів, яка не входить у навчальний набір як підмножина. Дану множину прикладів будемо називати тестовим набором. Для підвищення точності роботи мережі, додамо у процес навчання набір прикладів, який назвемо набір валідації. Будемо

використовувати його для локальної оцінки точності класифікації даних під час навчання.

2.2. Використання композиційної архітектури для навчання нейромереж

2.2.1. *Загальна ідея*

Одним із шляхів досягнення властивості адаптивності є використання композиційної структури. Основна ідея композиційності базується на використанні системи окремих та в основному незалежних підмереж, що працюють разом для вирішення загальної задачі (дана концепція будується на принципах нейробіології). У мозку є функціонально спеціалізовані області, які напрямлені на роботу із різними когнітивними процесами. Всередині частини мозку, яка називається таламусом, лежить бічне колінчасте ядро, яке розділене на шари, які окремо обробляють колір і контраст: обидва є основними компонентами зору. Композиційні нейронні мережі використовують цю концепції для вирішення складних проблем штучного інтелекту. Кілька незалежних нейронних мереж навчаються одночасно для реалізації конкретної підзадачі, а їх результати об'єднуються в кінці для виконання одного завдання. Переваги композиційних нейронних мереж включають:

- Простоту.
- Поєднання прийомів навчання.
- Масштабованість.
- Ефективність.

Ефективність даних архітектур полягає у реалізації властивості адаптивності. При наявності обмежених обчислювальних ресурсів, необхідно, щоб підмережі вихідної нейромережі були розподілені у відповідності із існуючими підзадачами. Вирішення кожної з підзадач має призводити до вирішення загальної задачі. Прикладом може слугувати

дрон, метою якого є розпізнавання об'єктів *А* та *Б*, які не мають спільних характеристик. При використанні композиційної архітектури, адаптивність буде набуватись при виконанні підзадач «розпізнавання об'єкту *А*» та «розпізнавання об'єкту *Б*» двома підмережами, отриманими з вихідної загальної мережі. Також адаптивність проявляється в тому, що кожну з підмереж можливо використовувати в рамках інших агентів без необхідності повного перенавчання. З точки зору обчислювальних ресурсів, даний підхід дозволяє виконувати навчання в реальному часі, так як загальна кількість операцій композиційної нейромережі є суттєво меншою за відповідну загальної мережі.

2.3. Існуючі методи навчання нейромереж на основі композиційних структур

Розглянемо декілька існуючих методів використання композиції при роботі із нейронними мережами. Загалом, дослідження композицій поділились на два напрями:

- Створення наборів даних для навчання та тестування нейронних мереж, які б стимулювали поведінку мережі як композиційного алгоритму. Використання нейросимволізму.
- Оцінка структури мереж, отриманих з навчених на незалежних наборах даних моделей.

2.3.1. *Методи, що включають в себе синтез наборів даних. Нейросимволізм*

Арифметична мова

Veldhoen et al. (2016) [15] запропонував в якості набору даних арифметичну мову, яка б дозволила дослідити, яким чином нейронні мережі оброблюють ієрархічно структуровані дані. Вони вводять таке поняття, як діагностичні класифікатори, демонструючи, що рекурентні нейронні мережі не мають суттєвого успіху при виконання таких завдань.

Експерименти також показують, що, при цьому, вентильні рекурентні нейронні мережі можуть краще узагальнювати довжину та глибину арифметичних виразів, які не були присутні в наборі даних для тренування.

Пізніше Saxton et al. (2019) [16] продемонстрував, що трансформери узагальнюють інформацію краще, ніж LSTM-моделі на тестах, які вимагають композиційних властивостей.

SCAN

Lake та Baroni у 2018 [17] році запропонували набір даних на основі навігаційної задачі. Задачею агента є виконання команд, сформованих композиційною мовою. Тестування моделей відбувалось на трьох наборах даних. Перший включав в себе випадкові приклади, другий – довші послідовності дій, третій – приклади, класифікація яких вимагає композиційних властивостей. Моделі надавали високу точність на першому наборі, на другому та третьому точність була низькою. Пізніше в публікації Loula et al. (2018) [18], автори запропонували новий підхід формування наборів даних. В 2019 Dessi та Baroni [20] випустили публікацію із діагностикою згорткових мереж, порівнявши результати із рекурентними. Було продемонстровано, що помилки, які допускають згорткові мережі не є систематичними. Тим не менш, мережі краще адаптували композиційні та узагальнюючи властивості.

Пошукові таблиці

Liska et al. (2018) [21] запропонували мінімальний композиційний тест, де мережа має використати функціональні композиції, щоб віднайти правильне значення послідовності таблиць. Значення даних таблиць подають на вхід мережі. Автори продемонстрували, що тільки невелика кількість моделей набуває композиційних властивостей.

Локальний пошук

Bowman et al. (2015) [22] запропонували оцінювати композиційні властивості моделі, тестуючи їх спроможність приходити до логічного

слідування (logical entailment). Сформована граматика включила короткі та прості речення. Задачею мережі було використати співвідношення між реченнями і за допомогою логічних обчислень отримати узагальнені вирази. Рекурентні нейромережі, що використовувались в експериментах, продемонстрували високу точність на даних наборах.

Mul and Zuidema (2019) [23], а також Tran et al. (2018) [24] продемонстрували подібні результати в своїх статтях – рекурентні мережі показують себе найкраще при набутті базових композиційних та узагальнюючих властивостей.

Композиційні аспекти

Hupkes et al. (2020) [25] запропонували дослідити різні аспекти композиційних властивостей, поділивши їх на п’ять категорій:

- Систематичність – комбінація відомих даних для отримання нових послідовностей.
- Продуктивність – можливість моделей отримувати довші послідовності у порівнянні із тими, на яких вони були натреновані.
- Підстановка – можливість моделей ідентифікувати та підставляти синоніми.
- Локалізм – локальність композиційних операцій (чи розраховуються менші складові перед більшими).
- Узагальнення – можливості моделі адаптувати обчислення у відповідності із правилами (чи одразу модель реалізує правила, який об’єм даних необхідний для того, щоб модель змогла їх реалізувати).

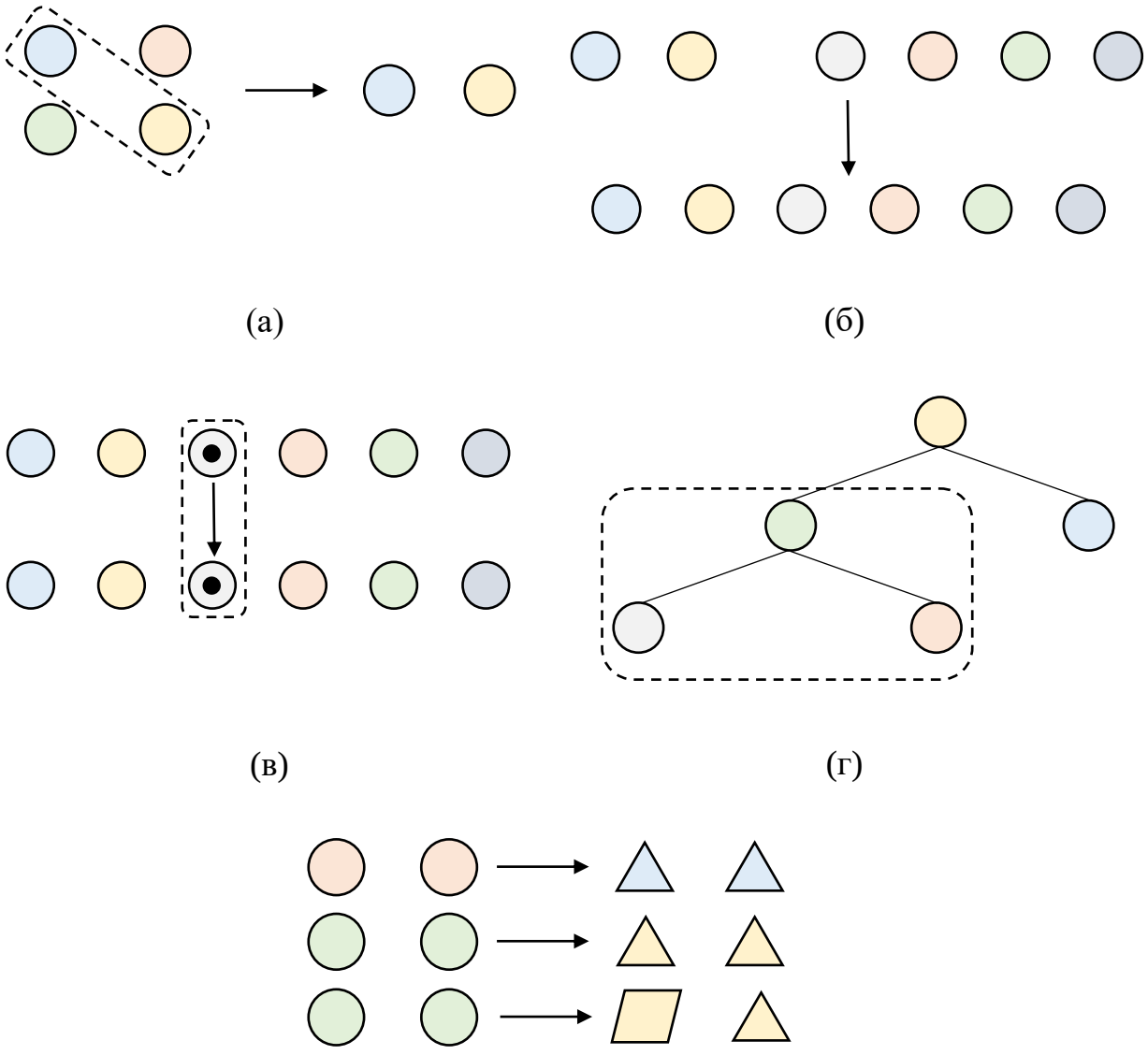


Рис. 2.1. П’ять композиційних стратегій: (а) систематичність, (б) продуктивність, (в) підстановка, (г) локалізм, (д) узагальнення

В якості моделей автори використовували LSTMS2S [26], ConvS2S [27], Transformer [28]. Результати показали, що жодна з архітектур не надає успішної реалізації систематичності (те саме стосується продуктивності). Говорячи про підстановку, Transformer та ConvS2S отримали гарні результати, в той час як робота LSTMS2S видалась менш успішною. Тестування моделей на локалізмі надало

результати навколо 50% точності. При тестуванні узагальнення жодна з моделей не показала успішних результатів.

Нейросимволізм

Дослідники гібридизують глибокі мережі з тим, що називають «старим штучним інтелектом», інакше відомим як символічний ШІ. Нинішні нейронні мережі здатні реалізовувати представлення логіки, немонотонного логічного програмування, модальної логіки та фрагментів логіки першого порядку (але не повноцінної логіки першого або вищих порядків) [29]. Дане обмеження спричинило початок дослідження в галузі логічних тензорних мереж (LTN) [30, 31, 32], яка для того, щоб використовувати мову повної логіки першого порядку при глибокому навчанні, переводить логічні твердження у функцію втрат, не маючи при цьому впливу на архітектуру мережі. Отже, логічні твердження першого порядку відображаються на диференційованих обмеженнях з дійсним значенням, використовуючи багатозначну інтерпретацію логіки в інтервалі від 0 до 1. Навчена мережа та логіка стають комунікаційними модулями гібридної системи, замість того, щоб логічні обчислення реалізовувались мережею.

2.3.2. *Методи, що включають в себе структурні композиції підмереж навчених на незалежних наборах даних моделей*

Модульні нейромережі

Результат структурної композиції декількох мереж в єдину також називають модульною нейронною мережею.

Sotirov et al. (2018) [33] описали подібний підхід при використання інткрементного аналізу та нечіткої логіки.

Ke Chen (2015) [34] у своїй роботі описав використання модульної архітектури для надання експертних оцінок в рамках задачі.

Даний напрямок заслуговує більшої уваги та розвитку. Саме тому в рамках даної дисертації досліджуються шляхи отримання композиційних (модульних) моделей нейромереж, з метою оптимізації процесу навчання.

Багатозадачне навчання

Багатозадачне навчання – це навчальна парадигма, в якій моделі машинного навчання навчаються з даними з кількох завдань одночасно, використовуючи спільні представлення, для вивчення загальних ідей між набором пов’язаних завдань. Ці спільні представлення підвищують ефективність передачі даних і потенційно можуть забезпечити більш швидку швидкість навчання для відповідних або подальших завдань, допомагаючи полегшити загальновідомі слабкі сторони глибокого навчання: великомасштабні вимоги до даних та обчислювальний попит. Однак досягнення таких ефектів виявилося непростим і є активною сферою досліджень сьогодні.

Існуючі методи багатозадачного навчання часто розділяються на дві групи: жорсткий обмін параметрами та м'який обмін параметрами. Спільне використання параметрів – це практика розподілу вагових коефіцієнтів між кількома завданнями так, що кожна вага обробляється для спільного мінімізації функцій втрат. За умови м’якого обміну параметрами різні завдання мають окремі моделі, специфічні для конкретного завдання, з окремими вагами, але різниця між параметрами моделі різних завдань додається до спільної цільової функції. Хоча не існує явного спільного використання параметрів, існує стимул для моделей, що відповідають конкретним завданням, мати подібні параметри.

2.4. Використання композиційних моделей для навчання нейромереж

Нехай P – композиція мереж. Тоді P_i – i-та підмережа даної композиції. Нехай $v_{P_I}^{(I)}$ – приклад, який надається мережі P_i для класифікацію, $v_{P_i}^{(O)}$ – результат класифікації приклада $v_{P_I}^{(I)}$ мережею P_i.

2.4.1. *Паралельна композиція*

Дана композиція працює за принципами композиційної властивості *продуктивність*, яка була описана в пункті 1.4.1.

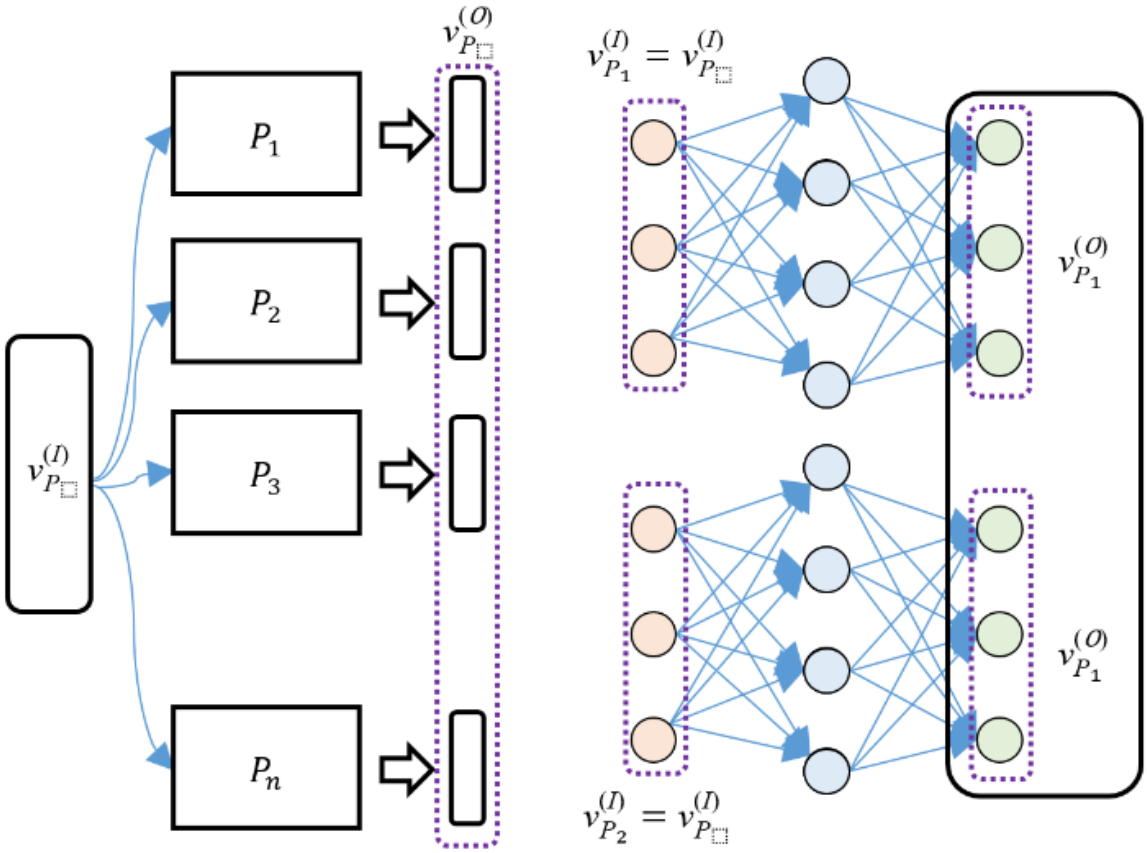


Рис. 2.2. Схема паралельної композиції. Результуючий вектор класифікації обчислюється як $concatenate(v_{P_1}^{(O)}, v_{P_2}^{(O)})$

Декілька нейронних мереж незалежно навчаються класифікувати різні компоненти вихідного вектора ознак. Тестовий приклад подається кожній з мереж, результати класифікацій конкатенуються. При оцінці точності

результат конкатенації порівнюється із вихідним вектором. Навчання відбувається в рамках одного набору даних.

Дана композиційна архітектура реалізовує властивість адаптивності при використанні підмереж для вирішення відповідних підзадач незалежно одна від одної. Даний підхід дозволяє як скоротити кількість обчислювальних операцій, так і використовувати кожну з підмереж для вирішення аналогічної підзадачі в рамках інших систем. Таким чином, кожна з підмереж стає взаємозамінним модулем.

2.4.2. *Послідовна композиція*

Дана композиція працює за принципами композиційної властивості *узагальнення* (пункт 1.4.1), яка реалізує послідовну докласифікацію вектора ознак кожною підмережою.

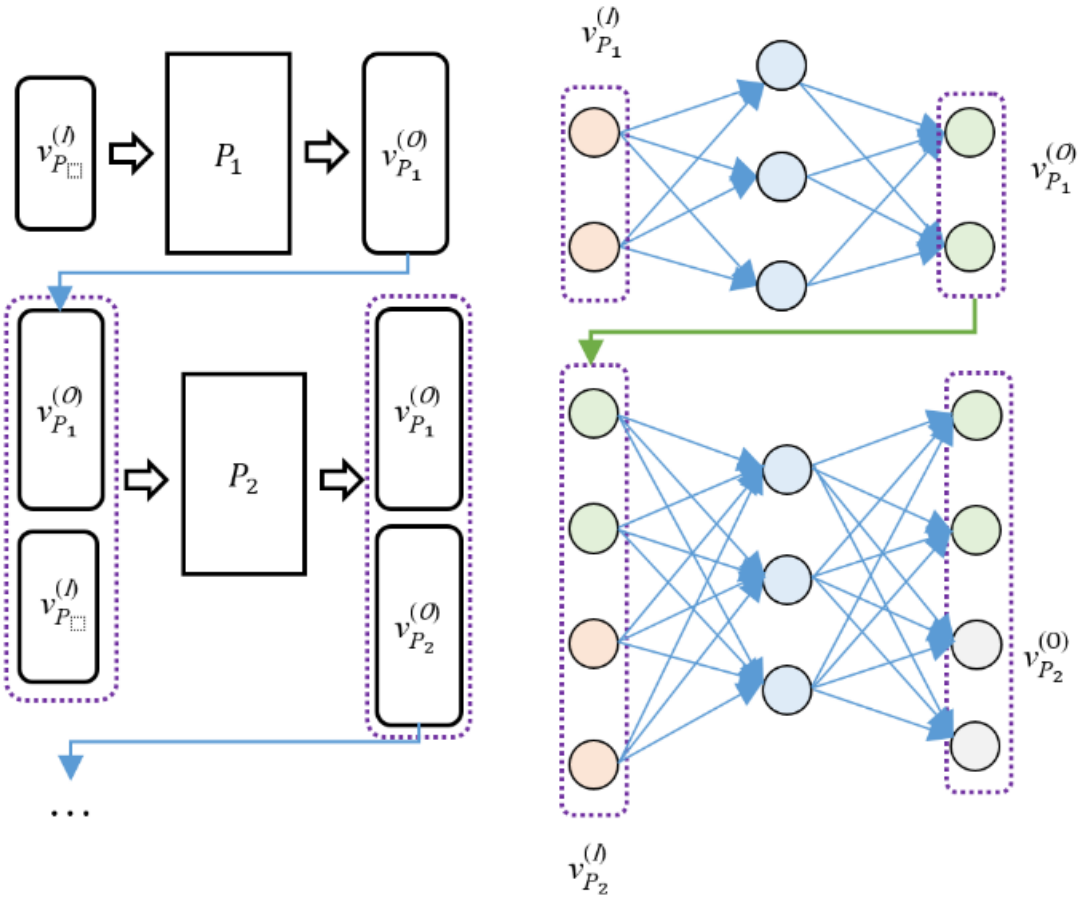


Рис. 2.3. Послідовна композиційна схема. Результуючим вектором класифікації є вектор $v_{P_2}^{(O)}$

Нейронні мережі незалежно навчаються класифікувати різні компоненти вихідного вектора ознак. При цьому, кожна наступна підмережа в якості вхідних даних отримує не тільки приклад, а й результати класифікації попередньої підмережі. Навчання відбувається в рамках одного набору даних.

Дана композиційна архітектура реалізовує властивість адаптивності при використанні підмереж, які залежать одна від одної.

Прикладом задачі, в рамках якої можливо використовувати даний підхід може слугувати: «розпізнати об'єкти *А* та *Б* такі, що об'єкт *Б* володіє спільними характеристиками із об'єктом *А* або залежить від нього». Замість використання загальної нейромережі адаптивність композиційної архітектури мережі дозволяє першій підмережі розпізнати об'єкт *А* та надати резальтати розпізнавання другій підмережі для розпізнання об'єкту *Б*. Дана реалізація дозволяє як використовувати зв'язки підмереж в рамках інших задач, так і робить підмережі взаємозамінними при використанні їх в різних агентах за умови, що вхідні дані будуть мати відповідний формат та інформацію.

2.5. Існуючі методи верифікації нейромереж

Розглянемо декілька існуючих методів верифікації нейронних мереж, що були розроблені.

2.5.1. *SMT-підхід*

Даний метод був розроблений вченими Стенфордського університету (Стенфорд, США) та полягає у використанні модифікованого симплекс-методу, який включає в себе ReLU-обмеження [35].

Ідея полягає в тому, що набір вхідних даних (у статті – атомів) може бути вбудований у логічну структуру. Подальший алгоритм виконує логічне розповсюдження, логічне розбиття та логічне вирішення конфліктів (Boolean propagation, case-splitting, and Boolean conflict

resolution). Після цього, алгоритм кодує ReLU-операції, використовуючи диз'юнкції. В результаті, глибинна нейронна мережа з n ReLU вузлами трансформується у $2n$ під-проблем, кожна з яких – диз'юнкція атомів.

Кодування ReLU-операції полягає у створенні пари змінних (у статті – v^b та v^f) та виконанні операції ReLU(v^b, v^f) . v^b, де v^b – обернено-напрямлена змінна (backward-facing variable), що використовується для виділення зв'язку v та вузлів з попереднього шару мережі, v^f – прямо-напрямлена змінна (forward-facing variable), що використовується для виділення зв'язку v та вузлів з наступного шару мережі (рис. 2.4).

Алгоритм перевіряє кожну з безпечних властивостей, обумовлених проблематикою задачі та її обмеженнями. Якщо вихідний набір, згенерований нейронною мережею, перетинає хоча б одну з них, модель мережі може вважатись недостатньо навченою або непридатною для функціонування в поставлених задачею умовах.

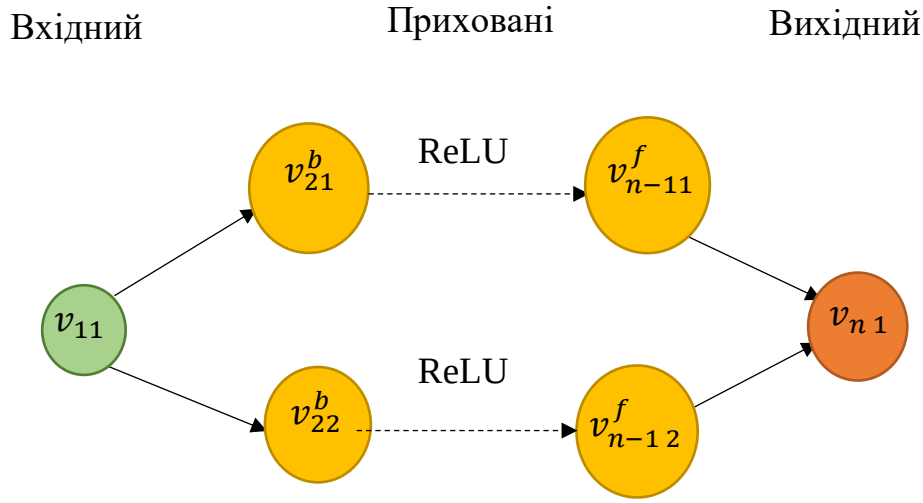


Рис. 2.4. Чотирьохшаровий перспептрон

2.5.2. *Алгоритм адаптивного уточнення та змагального пошуку*

Даний метод був розроблений вченими Імперського коледжу Лондона (Лондон, Англія) та полягає у використанні методу інтервального розповсюдження [36].

Проблема верифікації формулюється як набір:

$$< f, x_l, x_u, \varphi_y, L >, \text{ де } f : \mathbf{R}^m \rightarrow \mathbf{R}^n,$$

де x_l, x_u – вектори, що обумовлюють нижню та верхню границі вхідного набору даних, φ_y – набір лінійних обмежень вихідного набору даних, $L : \mathbf{R}^m \rightarrow \mathbf{R}$ функція нев'язки, що використовується для локального пошуку (рис. 2.5).

Алгоритм передає вирішує задачу лінійного програмування із вхідними обмеженнями x_l, x_u, та вихідними обмеженнями φ_y. Якщо рішення визначає, що проблема не задовольняє наданим обмеженням (множина рішень не перетинає множину, утворену обмеженнями), отримана модель є «безпечною» для використання у даній задачі.

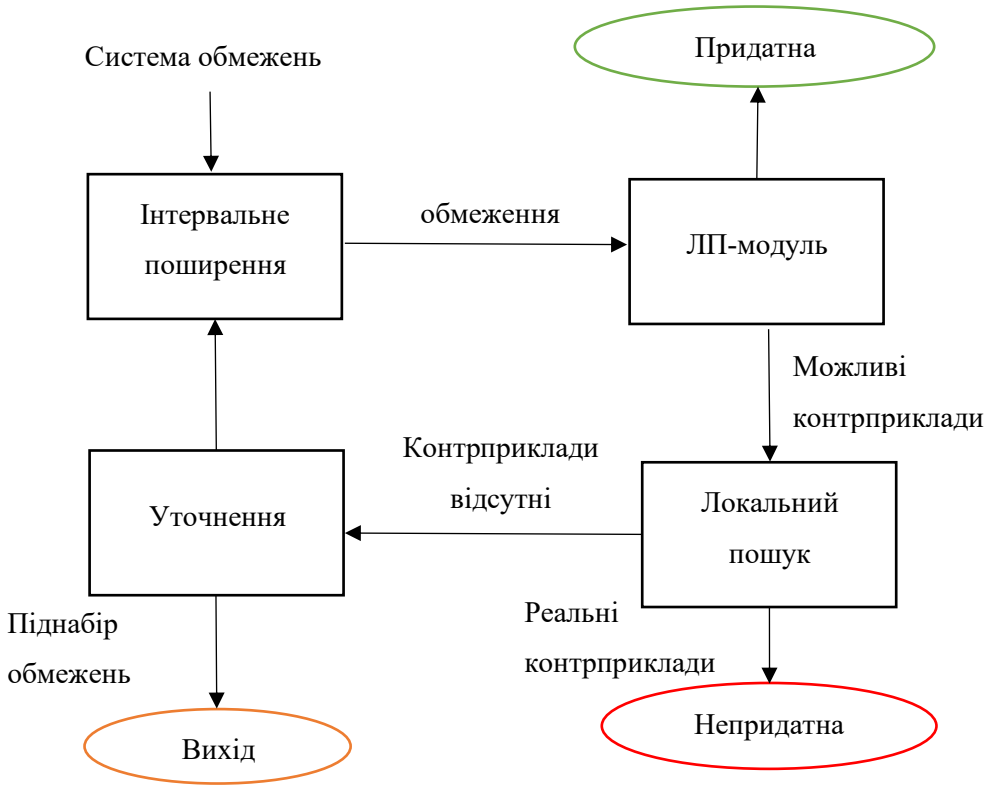


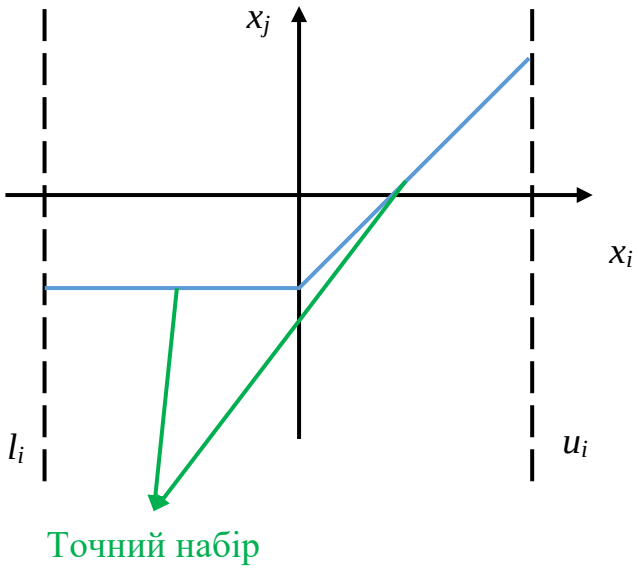
Рис. 2.5. Схема алгоритму

2.5.3. *Метод абстрактного домену*

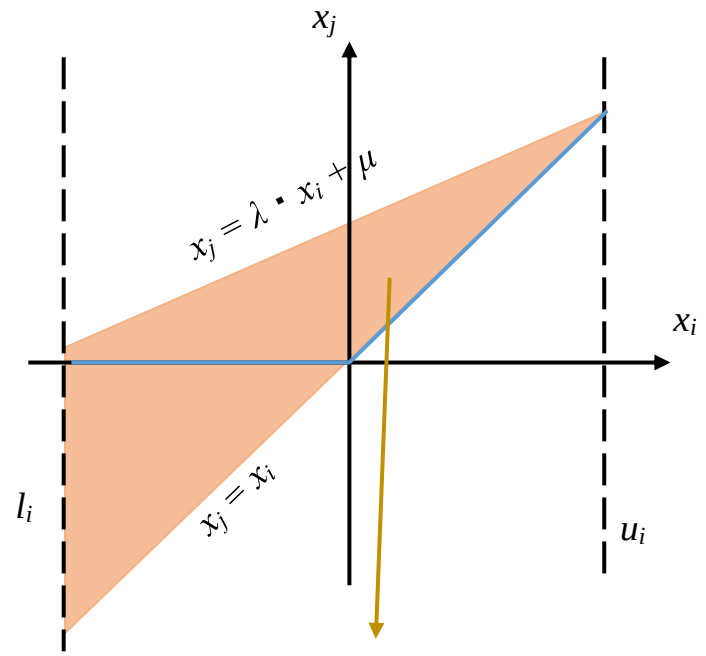
Даний алгоритм було розроблено вченими університету Цюриха (Цюрих, Швейцарія). Він полягає в апроксимації результатів обчислень кожного шару нейронної мережі прямого зв'язку [37].

Вихідний набір представляється у вигляді поліедрону, проходження через кожний шар нейронної мережі відповідає Афінному перетворенню ($W_{i+1,i} \times R_{Li} + b_{i+1}$, де $W_{i+1,i}$ – матриця вагових коефіцієнтів між шарами L_{i+1} та L_i, R_{Li} – результат обчислень попереднього шару, представлений апроксимацією об'єднання поліедронів, b_{i+1} – вектор біасу шару L_{i+1}), переданому у ReLU-операцію: ReLU($W_{i+1,i} \times R_{Li} + b_{i+1}$). Результат обчислень останнього шару відповідає ітоговому набору даним.

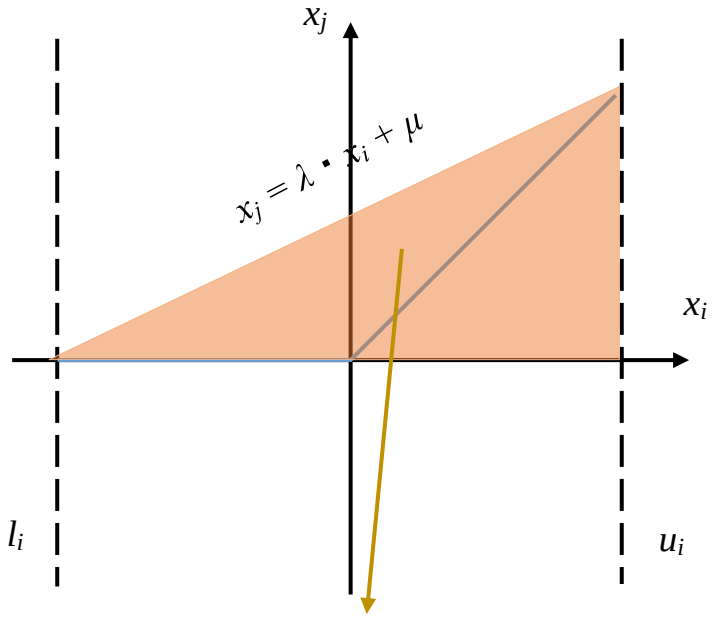
Припустимо, що неапроксимований набір поліедронів, отриманих внаслідок проходження через шар мережі виглядає, як на рис. 2.6 (а). Тоді алгоритм, описаний в статті пропонує наступні апроксимації (рис. 2.6).



(а)



(б)



(в)

Рис. 2.6. Апроксимації ReLU функції: (а) демонструє точний сет двох поліедронів, отриманий після обчислення одного шару персептрону, (б) та (в) демонструють дві опуклі апроксимації, представлені у статті

Змінні x_i та x_j представляють собою значення нейронів, що утворюють підмножину набору, отриманого в результаті обчислень. Змінні l та u відповідають значенням векторів нижньої та верхньої границі підмножини.

Апроксимація, що представлена на рис. 2.6 (б) накладає на змінні x_i та x_j наступні обмеження:

$$0 \leq x_j \leq u_i(x_i - l_i) / (u_i - l_i);\ l_j = 0,\ u_j = u_i \quad (1)$$

Апроксимація, що представлена на рис. 2.6 (в) накладає на змінні x_i та x_j наступні обмеження:

$$x_i \leq x_j \leq u_i(x_i - l_i) / (u_i - l_i);\ l_j = l_i,\ u_j = u_i \quad (2)$$

Набір, отриманий внаслідок апроксимації обчислень кінцевого шару мережі (об'єднання поліедронів), відповідає результату класифікації. Якщо набір перетинає властивість, встановлену як обмеження у формулюванні проблеми, нейронна мережа вважається непридатною до вирішення даної задачі класифікації.

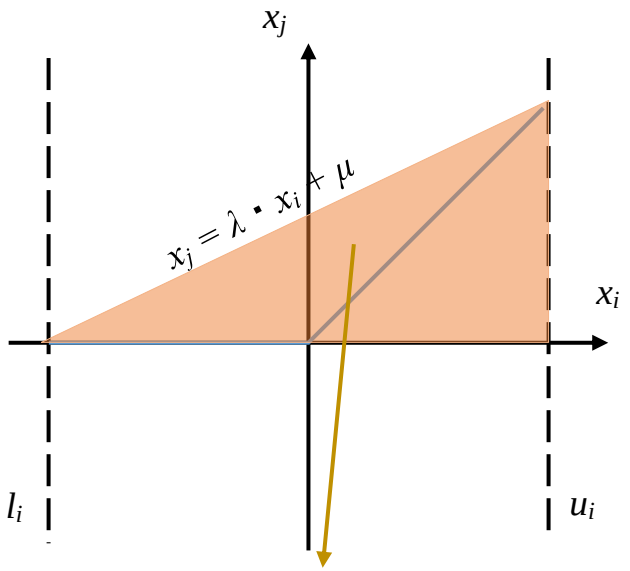
2.5.4. *Star set*

Даний алгоритм описує підхід, за допомогою якого можливо будувати як точний, так і апроксимований набір. Він був розроблений вченими Вандербілтського університету та університету Пенсильванії (США) [38].

"Star set" формально можна описати наступним чином: набір $S = < c, V, P >$, де $c \in \mathbf{R}^n$ – це центр, $V = \{v_1, v_2, \ldots, v_m\}$ є базисом в $\mathbf{R}^n$, $P(\alpha) \triangleq C\alpha \leq d$, де для p лінійних обмежень, $c \in \mathbf{R}^{p \times m}$, $\alpha = [\alpha_1, \ldots, \alpha_m]^T$, $d \in \mathbf{R}^{p \times 1}$.

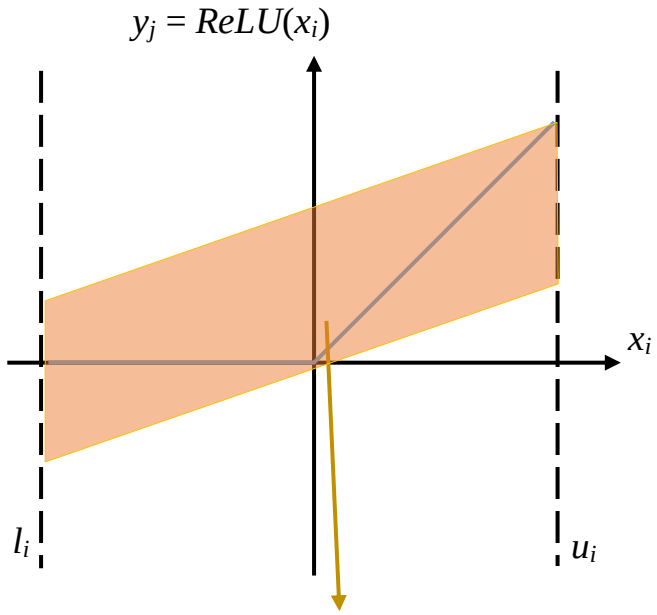
Алгоритм, описаний в статті, стверджує, що для шару нейронної мережі L, який має в собі n нейронів, результуючий набір можливо обчислити шляхом послідовного виконання n stepReLU-операцій: $\boldsymbol{R}_L = \mathrm{ReLU}_n(\mathrm{ReLU}_{n-1}(\ldots\mathrm{ReLU}_1(S)))..$, де S – "star set", обчислений попереднім шаром мережі.

Результат апроксимації, отриманий при використанні даного алгоритму є менш консервативним у порівнянні із апроксимацією зонотопом [39] або методом абстрактного домену (рис. 2.7).



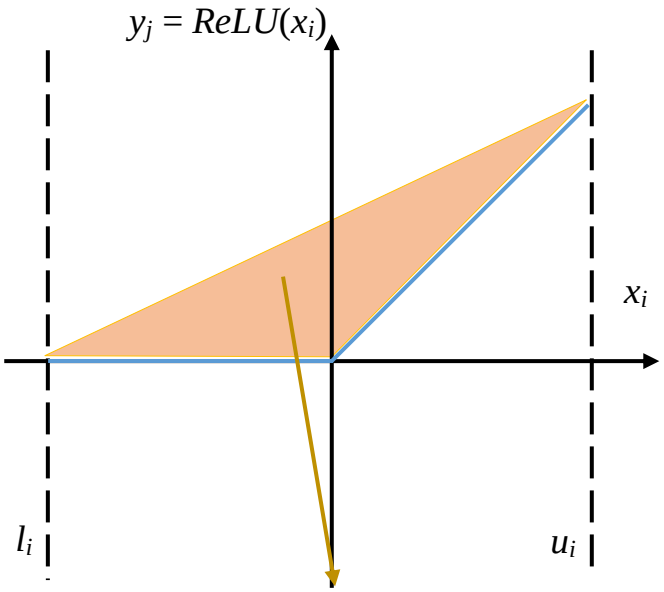
Апроксимація методом абстрактного

(а)



Апроксимований набір методом зонотопу

(б)



(в)

Рис. 2.7. Апроксимації ReLU функції: (а) демонструє апроксимований набір при використанні методу абстрактного домену, (б) демонструє апроксимований набір методом зонотопа, (в) демонструє апроксимований набір, використовуючи star-апроксимацію

Якщо кінцевий “star” набір має перетин із властивістю, встановлену як обмеження у формулюванні проблеми, нейронна мережа вважається непридатною до вирішення даної задачі класифікації.

2.6. Критерій оцінки навченості нейромереж

Сучасні методи верифікації нейронних мереж надають відповідь стосовно придатності (або непридатності) використання нейронної мережі для вирішення відповідної задачі класифікації як однозначну, не звертаючи на розподілення похибки на множині результуючого набору.

Результатом є можлива ситуація, в якій на результати нейронної мережі можна впливати ззовні шляхом подачі їй певних тестових прикладів, які будуть викликати високий рівень похибки при наданні результатів класифікації.

В якості завдання поставимо формування критерію оцінки рівня якості навченості нейронної мережі.

Задача на навчання нейромережі включає в себе: область визначення функції, необхідний рівень похибки, навчальний, валідуючий та тестовий набори прикладів. Встановивши граничне значення похибки, що вибирається для кожної з задач класифікації є постійним на всій області визначення функції, сформуємо наступні стани, кожний з яких описує ступень навченості мережі:

- Ненавчена нейронна мережа – мережа, що не проходила цикл тренування на наперед визначеному навчальному наборі даних. Кожна точка гіперплощини (в узагальненому випадку), що відповідає множині похибок, отриманих в результаті поставлення у відповідність тестовим прикладам класів, може бути поставлена у відповідність прикладу, в результаті класифікації якого її було отримано.
- Недонавчена нейронна мережа – мережа, що пройшла цикл тренування на наперед визначеному навчальному наборі даних, але точність класифікації якої не відповідає необхідній. Кожна точка гіперплощини (в узагальненому випадку) з множини похибок, отриманих в результаті поставлення у відповідність тестовим прикладам класів, відповідає меншій частині зони гіперплощини, обумовленої областю визначення функції, у порівнянні із зоною, що описана задачі класифікації.
- Навчена нейронна мережа – мережа, що пройшла цикл тренування на наперед визначеному навчальному наборі даних і точність класифікації якої відповідає необхідній. Кожна точка гіперплощини (в узагальненому випадку) з множини похибок, отриманих в результаті поставлення у відповідність тестовим прикладам класів, відповідає певній зоні гіперплощини, яка обумовлюється областю

визначення функції. Розмір та специфіка даної зони області визначаються формулюванням задачі класифікації.

- Перенавчена нейронна мережа – мережа, що пройшла цикл тренування на наперед визначеному навчальному наборі даних, але точність класифікації якої відповідає необхідній. Кожна точка гіперплощини (в узагальненому випадку), що відповідає множині похибок, отриманих в результаті поставлення у відповідність тестовим прикладам класів, відповідає певній точці зони гіперплощини, яка обумовлюється областю визначення функції. При цьому, кількість епох навчання є більшою, ніж поставлена в умові задачі класифікації.

Формулювання даного критерію дозволяє використовувати його для більш детального аналізу того, які піднабори даних викликають падіння точності розпізнавання. Це дозволяє оцінити, яким чином необхідно корегувати навчальний набір даних для підвищення якості роботи мережі. Даний підхід дозволяє зберегети об’єм ресурсів, який витрачається на навчання, так як на кожній наступній ітерації оптимізаційний процес швидше наближається до глобального оптимуму.

2.7. Висновки

У другому розділі розглянуті адаптивний метод навчання нейронних мереж на основі композиційної архітектури та критерій оцінки навченості нейромережі.

В результаті проведеного дослідження були розглянуті:

- Існуючі підходи до навчання нейромереж. Основна увага була приділена модифікації існуючих модульних архітектор.
- Існуючі підходи до верифікації нейромереж. Основна увага були приділена удосконаленню оціночного апарату методів, які використовують обчислення кінцевих наборів та їх перетин із безпечною властивості.

При розробці методу було сформовано:

- Альтернативний підхід до навчання нейромереж, який полягає у використанні композиційної архітектури мережі для однієї задачі. Метою описаних композицій є використання специфіки не тільки різних підчазач, але й різних ознак даних, які подаються на вхід. Дане формулювання дозволяє надати методу навчання властивість адаптивності. Окрім цього, стає можливим використання навчених підмереж для рішення аналогічних підзадач в рамках інших глобальних задач без необхідності перенавчання. Це дозволяє зберегти суттєву кількість обчислювальних ресурсів.
- Критерій оцінки навченості нейромережі, який дозволяє оцінити якість, використовуючи континуальну похибку. Похибка розраховується на континуальній множині прикладів, які формують набори даних для тренування, валідації та тестування. Це дозволяє використовувати його для більш детального аналізу того, які піднабори даних викликають падіння точності розпізнавання. Результатом є збереження обчислювальних ресурсів та, як наслідок, зменшення часу, який витрачається на навчання.

Для тестування даних концепцій була використана заздалегідь сформована модельна задача, описана в розділі 3. В якості типу нейромереж було обрано персептрон.

3. ОСОБЛИВОСТІ МОДЕЛЬНОЇ ЗАДАЧІ. ЕЛЕМЕНТИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1. Основні вимоги до модельної задачі та програмної реалізації

На основі поставлених задач оптимізації процесу навчання нейромереж та покращення оцінки їх навченості, а також описаних в другому розділі методу навчання нейромереж на основі композиційної архітектури та сформованого критерію навченості мережі, були сформовані наступні вимоги до модельної задачі та програмної реалізації:

Таблиця 3.1

Вимоги до модельної задачі

Код вимоги	Опис	Пріоритет
MR1	Формулювання задачі має забезпечувати доступні способи генерації наборів даних.	Високий
MR2	Розмірність задачі має бути масштабованою.	Високий
MR3	Формат даних має дозволяти виконувати навчання за доступну кількість часу (час навчання одного екземпляру мережі не має перевищувати 30 хв.).	Високий
MR4	Дані, що використовуються в рамках задачі, мають володіти характерними піднаборами ознак, які можливо виокремити.	Високий

Таблиця 3.2

Вимоги до програмної реалізації

Код вимоги	Опис	Пріоритет
I1	Дослідницький програмний комплекс має включати в себе п’ять основних модулів: • модуль генерації наборів даних; • модуль передобробки наборів даних; • модуль генерації конфігураційних файлів; • модуль навчання моделей; • модуль візуалізації та збереження результатів.	Високий
Модуль генерації наборів даних		
I2	Модуль генерації наборів даних має формувати набори в якості csv-файлів.	Високий
I3	Модуль генерації наборів даних має бути масштабованим з точки зору способу генерації. Він має дозволяти ефективно переключатись між законами розподілу генерованих прикладів.	Високий
I4	Програмне забезпечення, в якому розроблюється модуль має бути орієнтоване на роботу із великими наборами даних (від сотні тисяч записів). Бажано, щоб середа володіла спеціальними функціями роботи із списками.	Високий
I5	Програмне забезпечення, в якому розроблюється модуль має надавати робочий файл, який можливо запустити з іншого додатку.	Високий
Модуль передобробки наборів даних		
I6	Модуль має виконувати перевірку на автентифікацію прикладів.	Високий
I7	Програмне забезпечення, в якому розроблюється модуль має надавати робочий файл, який можливо запустити з іншого додатку.	Високий
Модуль генерації конфігураційних файлів		
I8	Модуль має виконувати генерацію конфігураційних файлів для наборів даних та моделей навчання.	Високий

Продовження табл. 3.2

I9	Формат конфігураційних файлів має підтримуватись автоматичними генераторами словників.	Високий
I10	Програмне забезпечення, в якому розроблюється модуль має надавати робочий файл, який можливо запустити з іншого додатку.	Високий
Модуль навчання моделей		
I11	Модуль має виконувати навчання, валідації та тестування моделей на основі наданих конфігураційних файлів наборів даних та моделей.	Високий
I12	Модуль має підтримувати збереження та завантаження моделей.	Високий
I13	Модуль має реалізовувати об’єктно-орієнтовні принципи для забезпечення легкого масштабування.	Високий
I14	Модуль має реалізовувати парсер конфігураційних файлів наборів даних та моделей, а також файлів із наборами даних.	Високий
I15.1	Модуль має підтримувати створення композиційних архітектур мереж.	Високий
I15.2	Модуль має реалізовувати можливість створення кластерів моделей (кожний кластер відповідає заданому типу композиції).	Високий
I15.3	Модуль має підтримувати збереження кластерів в файловій системі та їх завантаження.	Високий
I15.4	Модуль має реалізовувати техніки форматування наборів даних для роботи із кластерами в режимі реального часу.	Високий
I16	Модуль має підтримувати адаптивну структуру, побудовану на базі вказівників (реалізовувати окремі контейнери для наборів даних, моделей та кластерів нейромереж).	Високий

Продовження табл. 3.2

Модуль візуалізації та збереження результатів		
I17	Модуль має надавати візуалізацію похибки на контінуальній множині, використовуючи отримані результати класифікації.	Високий
I18	Модуль має зберігати результати класифікації у файлах csv-формату.	Високий

3.2. Опис модельної задачі

Модель, що використовується в даному дослідженні, формально може бути описана наступним чином:

Маємо гіперплощину A, та функцію $G(\mathbf{X})$, визначену наступним чином:

$$G(\mathbf{X}) \bowtie \mathbf{Y}, \tag{4}$$

де $\mathbf{X} \in \mathbf{R}^n$ – набір змінних, $G(\mathbf{X})$ – набір лінійних форм, $\mathbf{Y} \in \mathbf{R}^k$ – вектор-константа, $\bowtie$ інтерпретуємо як набір наступних операторів: $\{\leq, \geq, =\}$.

Тоді, модель можна представити наступним чином:

$$G\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} \lesseqgtr \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_k \end{bmatrix}. \tag{5}$$

З точки зору задачі класифікації, функція $G(\mathbf{X})$ та вектор констант $\mathbf{Y}$ представляють собою набір $[D_{train}; D_{cv}; D_{test}]$. D_{train} є навчальним набором, D_{cv} є валідуючим набором, D_{test} є тестовим набором. Набір операторів, елементи якого необхідно поставити у відповідність кожній з пар, відповідає набору класів C. Оператор $\leq$ еквівалетний значенню “0” у векторі ознак, оператор $\geq$ еквівалетний значенню “1” у векторі ознак, оператор $=$ еквівалетний значенню “2” у векторі ознак.

Для підвищення зручності візуалізації та спрощення процесу обчислювань, зведемо розмірність простору до двох. Тоді, отримана модель буде мати наступне формулювання:

$$G\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leqslant \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_k \end{bmatrix}. \tag{6}$$

Так як компоненти вектора $G(\mathbf{X})$ представляють собою лінійні форми, в рамках двомірного простору задача зводиться до локалізації областей, на які прямі поділяють площину в залежності від точки, що поступає на вхід.

3.3. Опис використаних нейронних мереж та функцій похибки

В якості нейронної мережі для проведення експериментів, було обрано персептрон, так як даний тип мереж дозволяє зменшити індивідуальний та загальний час виконання операцій у порівнянні з іншими архітектурами для обчислення простих задачах. Кількість епох, на яких було виконане навчання – 50, 150. Набори даних, що використовувались для навчання та тестування поділялись на: ті, що мають приклади для чотирьох обмежень, ті, що мають приклади для восьми обмежень. Кількість прикладів в кожному з наборів становить 128 тис. В якості розподілу прикладів використовувався нормальний розподіл. В якості функції оцінки похибки було використане середньоквадратичне відхилення результату класифікації від даного значення:

$$\varepsilon^{(i)}\left(p^{(i)}, c^{(i)}\right) = \frac{\sum_{j=1}^{k}\left(p_j^{(i)} - c_j^{(i)}\right)^2}{k}. \tag{7}$$

Було сформовано три екземпляри мережі, з якими відбувалось порівняння:

- Персептрон із двома прихованими шарами наступних розмірностей: $||L_1|| = 16$, $||L_2|| = 8$.

- Персептрон із трьома прихованими шарами наступних розмірностей: $||L_1|| = 8$, $||L_2|| = 16$, $||L_3|| = 8$.
- Персептрон із шістьома прихованими шарами наступних розмірностей: $||L_1|| = 8$, $||L_2|| = 16$, $||L_3|| = 32$, $||L_4|| = 32$, $||L_5|| = 16$, $||L_6|| = 8$.

Відповідно, були сформовані наступні підмережі для обох варіантів композицій (паралельної та послідовної):

- Композиції, що включили в себе по два персептрони із двома прихованими шарами наступних розмірностей: $||L_1|| = 8$, $||L_2|| = 4$.
- Композиції, що включили в себе по два персептрони із трьома прихованими шарами наступних розмірностей: $||L_1|| = 4$, $||L_2|| = 8$, $||L_3|| = 4$.
- Композиції, що включили в себе по два персептрони із шістьома прихованими шарами наступних розмірностей: $||L_1|| = 4$, $||L_2|| = 8$, $||L_3|| = 16$, $||L_4|| = 16$, $||L_5|| = 8$, $||L_6|| = 4$.

Перевагою композицій над одиничними моделями є кількість обчислень. Якщо N_0 – кількість шарів одиничної мережі, n_i – кількість нейронів у шарі i одиничної мережі; $N^{(j)}$ – кількість шарів підмережі j, $m_i^{(j)}$ – кількість нейронів у шарі i підмережі j, тоді, не враховуючи застосування функції активації та додання біасу:

- Кількість операцій, яку необхідно виконати при обчисленні шару L_i одиничної мережі: $n_i \cdot n_{i+1}$.
- Кількість операцій, яку необхідно виконати при обчисленні шару L_i підмережі j: $m_i^{(j)} \cdot m_{i+1}^{(j)}$.
- Кількість операцій, яку необхідно виконати при прямому розповсюдженні в одиничній мережі:

$$O_0 = \sum_{i=1}^{N_0-1} n_i \cdot n_{i+1} \tag{8}$$

- Кількість операцій, яку необхідно виконати при прямому розповсюдженні в підмережі j:

$$O^{(j)} = \sum_{i=1}^{N^{(j)}-1} m_i^{(j)} \cdot m_{i+1}^{(j)} \quad (9)$$

- Вираз (9) не буде більшим за величиною, ніж вираз (8) при $N^{(j)} \le N_0$ та $m_i^{(j)} \le n_i$.

В нашому випадку $N^{(j)} = N_0$, $m_i^{(j)} = \frac{n_i}{2}$. Тоді:

$$\frac{O_0}{O^{(j)}} = \frac{\sum_{i=1}^{N_0-1} n_i \cdot n_{i+1}}{\sum_{i=1}^{N_0-1} \frac{n_i}{2} \cdot \frac{n_{i+1}}{2}} = 4.$$

Для опису результатів, введемо наступне умовне позначення для ідентифікації нейромежі: персептрон зі структурою прихованих шарів P, навчений на i епохах, на наборі даних із j обмеженнями та параметром сітки k запишемо як $P\_i\_j\_k$. Значення параметра k обумовлює прямокутник, який вміщує в себе множину прикладів. Формально він визначається наступним чином:

$$S = \{p_i \mid p_i \triangleq \{-k \le x_i \le k\},\ i = 1 \ldots n\} \quad (10)$$

3.4. Опис програмної реалізації

Розглянемо детально реалізацію кожного з модулів, які були реалізовані в рамках описаного програмного комплексу для тестування запропонованих адаптивного методу навчальних мереж та критерію навченості.

3.4.1. *Модуль генерації наборів даних*

Даний модуль було розроблено програмними засобами математичного пакету Maxima v. 5.44.

Maxima – це система для маніпулювання символьними та числовими виразами, включаючи диференціювання, інтегрування, ряд Тейлора, перетворення Лапласа, звичайні диференціальні рівняння, системи

лінійних рівнянь, поліноми, множини, списки, вектори, матриці та тензори. Maxima дає високоточні числові результати, використовуючи точні дроби, цілі числа довільної точності та числа з плаваючою комою із змінною точністю. Maxima може будувати функції та дані у двох та трьох вимірах.

Вихідний код Maxima може бути скомпільований у багатьох системах, включаючи Windows, Linux та MacOS X. Вихідний код для всіх систем.

Maxima – нащадок Macsyma, системи комп'ютерної алгебри, розробленої наприкінці 1960-х років у Массачусетському технологічному інституті. Це єдина система, яка все ще є загальнодоступною та має активну спільноту користувачів, завдяки своїй природі з відкритим кодом.

Вибрів даного програмного додатку пов'язаний із вимогою І4. Будучи написаною на діалекті Lisp, Maxima надає широкий набір можливостей обробки даних, які зберігаються у списках. Даний факт дозволяє як зменшити час виконання обчислювальних операцій, так і зробити кодову базу більш компактною та доступною для читання.

Приклади двох функцій для формування вектору ознак, формування списку точок, а також формат файлу із набором даних наведені нижче:

Лістинг 1. Функції для формування вектору ознак

```
checkPresence(currentFeature) := (
    if assoc(currentFeature[FEATURE_VECTOR_ID], areaClasses) = false
        then (basicPoints : append(basicPoints,
[currentFeature[POINT_ID]]),
                composeArea(currentFeature[FEATURE_VECTOR_ID]),
                basicFeature : append(basicFeature,
[[append(currentFeature[POINT_ID],
[currentFeature[FEATURE_VECTOR_ID]])]]),
                initPoints : append(initPoints,
[currentFeature[POINT_ID]]),
                testFeature : append(testFeature,
[[append(currentFeature[POINT_ID],
[currentFeature[FEATURE_VECTOR_ID]])]]))
```

```
        else (initPoints : append(initPoints,
[currentFeature[POINT_ID]]),
                testFeature : append(testFeature,
[[append(currentFeature[POINT_ID],
[currentFeature[FEATURE_VECTOR_ID]])]]))
) $
```

Лістинг 2. Функції для формування вектору ознак

```
tracePoints(testLines, pointsNumber, parameters) := block (
    set_random_state (true),
    pointsList : [],
    if parameters[PREDEFINED_FLAG] = 1
       then (pointsList : read_nested_list("predefined.txt",
semicolon)),
     if parameters[RANDOM_FLAG] = 1
       then (pointsList : append(pointsList,
create_list([random(RIGHT_THRESH + abs(LEFT_THRESH)) + LEFT_THRESH,
                random(RIGHT_THRESH + abs(LEFT_THRESH)) +
LEFT_THRESH], i, 1, pointsNumber))),
     if parameters[SAVE_PRREDEFINED_FLAG] = 1
          then (predefined : append(predefined, pointsList)),
    map(lambda([point], checkPresence([point, composeFeature(point,
testLines)])), pointsList)
) $
```

Лістинг 3. Формат вхідних даних

```
-0.1721617548795058;0.06892889305884187;"1100"
-0.2416641874037959;0.3588118173763075;"1010"
-0.2821443827442756;-0.3947953331590259;"1100"
-0.283001089254667;-0.07616106859255334;"1100"
0.208729562098839;0.423639144863406;"1011"
-0.1098099028702364;0.4716231507498931;"1011"
-0.4998436414919354;0.1411825949062817;"1200"
-0.08178547851218587;0.1430159810390295;"1000"
...
```

3.4.2. *Модулі передобробки наборів даних та генерації конфігураційних файлів*

Дані модулі були розроблені при використанні мови Python стандарту 3.7.

Причиною використання Python є його гнучкість та простота з точки зору написання коду. Ще однією причиною є те, що ядро комплексу (модуль навчання моделей) було також розроблено мовою Python.

Модуль передобробки наборів даних проходить по прикладах з відповідного набору, шукаючи співпадіння. Знайшовши співпадіння, модуль видаляє приклад та генерує новий із подальшою перевіркою.

Модуль генерації конфігураційних файлів генерує два види конфігурацій – для нейромереж та для наборів даних.

Конфігураційний файл нейромережі представляє собою набір ключових атрибутів та їх значень, які після подання в модуль навчання, оброблюються та ініціалізують відповідні атрибути класів.

Приклади конфігураційних файлів для набору даних та нейромережі наведені нижче:

Лістинг 4. Формат файлу конфігурації для набору даних

```
"path":"./GeneratedDatasets//128000_2_2_-0.25_0.25_1/
negativeDir128000_2_2_-0.25_0.25_1.txt"
"masks":"[[1,1,0,0]]"
```

Лістинг 5. Формат файлу конфігурації для нейронної мережі

```
"nn_type" : 0
"optimizer_id" : 1
"learning_rate" : 0.01
"learning_rate_decay_flag" : 0
"epochs" : 150
"mini_batch_size" : 128
"output_activation_id" : 1
"callback_ids" : [0, 1]
"log_path" : "logs/fit/"
"metrics" : [0, 1, 2]
"hidden_layers" : {
```

```
        1 : {
                "nodes_num" : 8,
                "kernel_reg_rate" : None,
                "bias_reg_rate" : 0.1,
                "hidden_activation" : 0
        },
        2 : {
                  "nodes_num" : 16,
                  "kernel_reg_rate" : None,
                  "bias_reg_rate" : 0.1,
                  "hidden_activation" : 0
         },
         3 : {
                  "nodes_num" : 32,
                  "kernel_reg_rate" : None,
                  "bias_reg_rate" : 0.1,
                  "hidden_activation" : 0
         },
         4 : {
                  "nodes_num" : 32,
                  "kernel_reg_rate" : None,
                  "bias_reg_rate" : 0.1,
                  "hidden_activation" : 0
         },
         5 : {
                  "nodes_num" : 16,
                  "kernel_reg_rate" : None,
                  "bias_reg_rate" : 0.1,
                  "hidden_activation" : 0
         },
         6 : {
                  "nodes_num" : 8,
                  "kernel_reg_rate" : None,
                  "bias_reg_rate" : 0.1,
                  "hidden_activation" : 0
         }
}
```

3.4.3. *Модуль навчання моделей*

Даний модуль було розроблено при використанні мови Python стандарту 3.7. Він є ключовим у всьому комплексі і виконує тренування моделей та кластерів моделей, включаючи збереження на диск та завантаження.

В рамках модулю було реалізовано наступні класи, описані в таблиці 3.3.

Таблиця 3.3

Список класів модулю навчання

Назва	Опис
BaseModel	Базовий клас моделі нейронної мережі. Зберігає чисельні значення гіперпараметрів для навчання. Виконує ініціалізацію моделі, її навчання та передбачення, збереження та завантаження.
BasicCluster	Представляє собою кластер моделей і реалізує композиційні властивості. Виконує навчання та передбачення композиційної архітектури, яка реалізована всередині кластера.
BasicContainter	Представляє собою контейнер, через яких відбувається робота із моделями та кластерами. Виконує ініціалізацію, навчання та передбачення вибраних моделей або кластерів.
BasicScope	Представляє собою область збереження наборів даних.
Dataset	Представляє собою сутність набору даних. Зберігає в собі ключові слова атрибутів, методи для завантаження та збереження набору даних. Реалізовує алгоритм маскування та розподілу доступу до навчального, валідуючого і тестового наборів в рамках наданого набору прикладів.
DefaultParser	Представляє собою сутність парсера для всіх конфігураційних файлів.
Perceptron	Є спадкоємцем класу BaseModel. Реалізовує в собі структуру персептрону.

Діаграма класів модулю представлена на рис. 3.1.

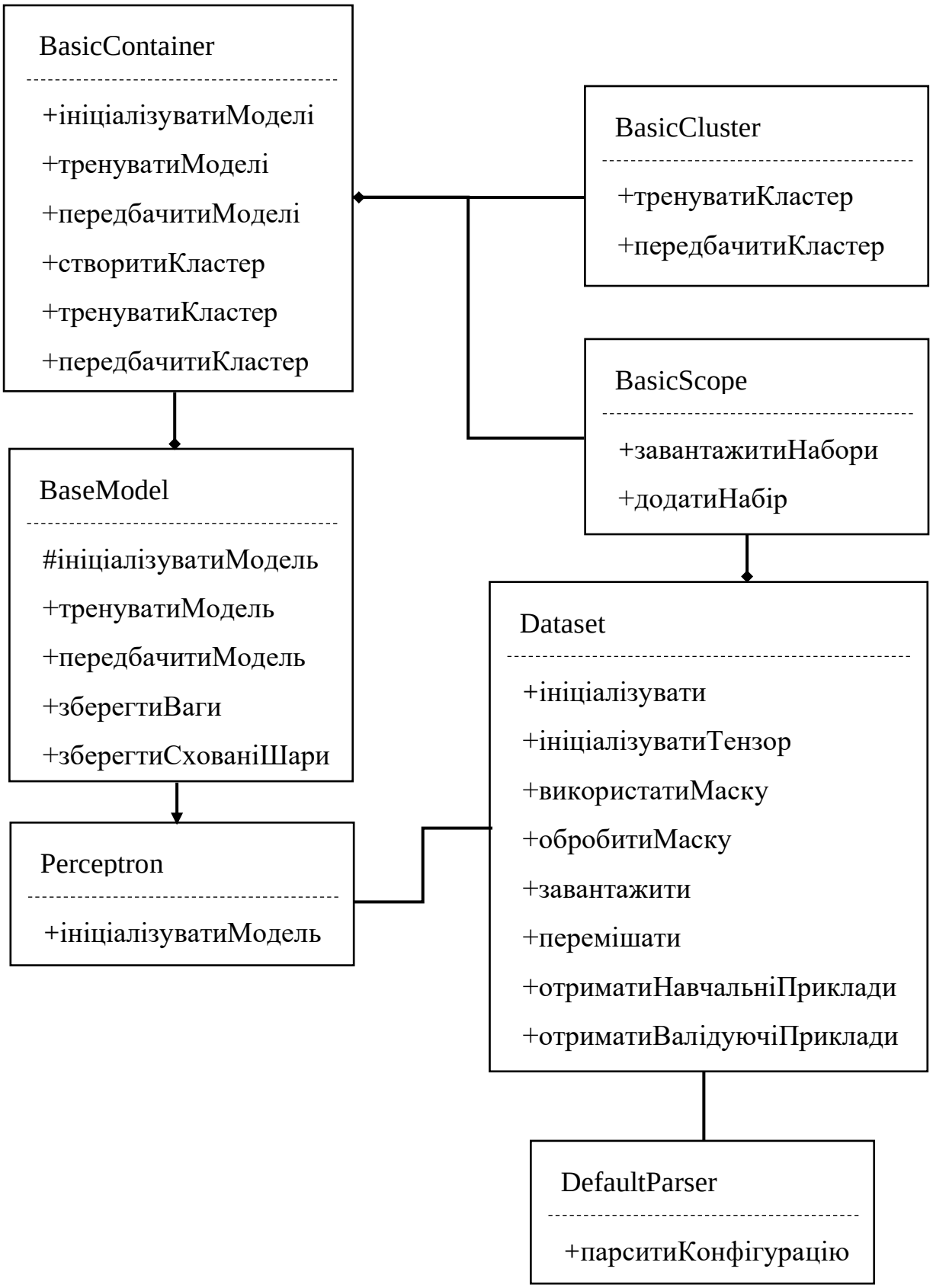


Рис. 3.1. Діаграма класів модулю навчання моделей

Нижче наведені приклади функції запуску тренування моделей класу BasicContainer та функції тренування моделі класу BaseModel (лістинг 6).

Лістинг 6. Функція тренування кластерів нейронних мереж

```
def trainModels(self, datascope = None, assignmentDict = None):
        if assignmentDict is None:
            for dataset in datascope.getDatasets():
                X_train, y_train, X_cv, y_cv, X_test, y_test =
dataset.unload()

                for currentModel in self.modelsList:
                    currentModel.initSession(y_train.shape[1],
X_train.shape[1], y_train.shape[1])
                    currentModel.train(X_train, y_train, X_cv, y_cv,
X_test, y_test)
        else:
            for modelID in assignmentDict.keys():
                X_train, y_train =
datascope[assignmentDict[modelID][TRAIN_DS_ID]].unloadTrain(SHUFFLE_
OFF)
                X_cv, y_cv =
datascope[assignmentDict[modelID][CV_DS_ID]].unloadCV(SHUFFLE_OFF)
                X_test, y_test =
datascope[assignmentDict[modelID][TEST_DS_ID]].unloadTest(SHUFFLE_OF
F)


self.modelsList[modelID].initSession(y_train.shape[1],
X_train.shape[1], y_train.shape[1])
                self.modelsList[modelID].train(X_train, y_train,
X_cv, y_cv, X_test, y_test,
datascope[assignmentDict[modelID][TRAIN_DS_ID]].get_name())
```

Лістинг 7. Функція тренування одиничних нейронних мереж

```
def train(self, X_train, y_train, X_cv, y_cv, X_test, y_test,
ds_name):

        self.model.compile(optimizer =
optimizers[self.optimizerID](self.learning_rate),
loss = lossFunctions[self.lossFunctionID], metrics = self.metrics)

        tensorboard_callback = tf.keras.callbacks.TensorBoard(
log_dir=self.log_dir, histogram_freq=1, profile_batch = 0)

        saveModel_callback =
tf.keras.callbacks.ModelCheckpoint(filepath=self.log_dir,
save_best_only=True, monitor = 'val_acc', period = self.epochs_num)


        self.model.fit(X_train, y_train, epochs = self.epochs_num,
batch_size = self.batch_size, validation_data = (X_cv, y_cv),
callbacks=[tensorboard_callback, saveModel_callback])
```

```
        self.model.evaluate(X_test, y_test, batch_size =
self.batch_size)

        return self.model
```

3.4.4. *Модуль візуалізації та збереження результатів*

Даний модуль зберігає результати класифікації моделей та візуалізує їх у відповідності із контінуальною похибкою. Модуль реалізовано засобами Python 3.7, для візуалізації використовується пакет matplotlib.

Файли результатів мають наступний формат:

Лістинг 8. Формат вихідних даних

```
<координати_точки>; <вектор_передбачень>; <вектор_ознак>;
<значення_похибки>; <функція_похибки>;

[-0.14889 -0.22109]; [0.3354336 0.31667694 0. 0.34788948]; [1 1 0 1];
0.3334567553770811; mse.
```

Результати візуалізації представляють собою графіки наступного вигляду (кожна лінія відповідає обмеженню, кожна точка – прикладу, інтенсивність кольору точки – значенню похибки):

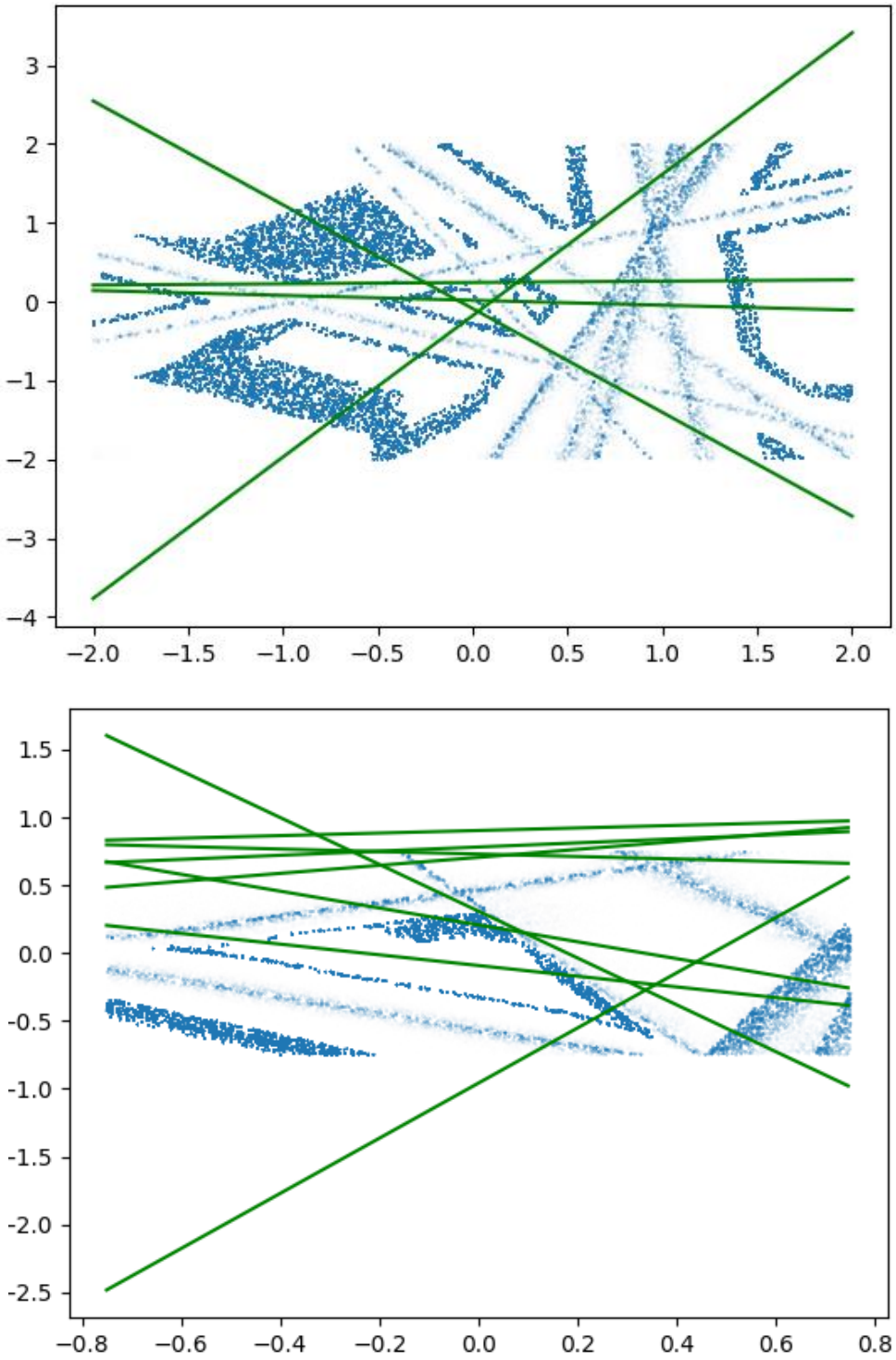


Рис. 3.2. Приклад візуалізації результатів класифікації

3.5. Висновки

В рамках дослідження було сформовано підхід структурної композиції при роботі із нейромережами та критерій навченості нейромережі. Навчені мережі було протестовано на відповідній модельній задачі, що представляє собою гіперплощину із рядом обмежень. В якості

класів використовуються набори операторів, кожен з яких обумовлює певну область, на які гіперплощина ділиться поставленими обмеженнями (експеримети, наведені у статті, використовують двовимірний простір). Було проведено дві стадії експериментів – без використання технології дропаута, та разом із нею.

Було реалізовано програмний комплекс засобами Python 3.7 та Maxima 5.44. Комплекс розроблювався відповідно до вимог, описаних в розділі 3.4.1 та включив в себе наступні модулі:

- Модуль генерації наборів даних.
- Модуль передобробки наборів даних.
- Модуль генерації конфігураційних файлів.
- Модуль навчання моделей.
- Модуль візуалізації та збереження результатів.

При розробці, були використані наступні бібліотеки:

- Tensorflow v. 2.0 – тензорна бібліотека машинного навчання.
- Keras v. 2.0 – окремий інтерфейс Tensorflow для роботі із моделями нейронних мереж.
- Matplotlib – бібліотека для побудови та візуалізації графіків.
- Numpy – бібліотека для мови програмування Python, що додає підтримку великих багатовимірних масивів та матриць, а також велику колекцію математичних функцій високого рівня для роботи з цими масивами.
- Scikit-Learn – безкоштовна програмна бібліотека машинного навчання для мови програмування Python. Він має різні алгоритми класифікації, регресії та кластеризації, включаючи машини векторної підтримки, тощо.

4. АНАЛІЗ ЕФЕКТИВНОСТІ МЕТОДУ АДАПТИВНОГО НАВЧАННЯ НЕЙРОМЕРЕЖ ТА РОЗРОБЛЕНОГО КРИТЕРІЮ ВЕРИФІКАЦІЇ

Для опису результатів, введемо наступне умовне позначення для ідентифікації нейромережі: персептрон зі структурою прихованих шарів *P*, навчений на *i* епохах, на наборі даних із *j* обмеженнями та параметром сітки *k* запишемо як *P_i_j_k*. Значення параметра *k* обумовлює прямокутник, який вміщує в себе множину прикладів. Формально він визначається наступним чином:

$$S = \{p_i \mid p_i \triangleq \{-k \le x_i \le k\},\ i = 1 \ldots n\} \tag{8}$$

4.1. Аналіз ефективності методу композиційного навчання

4.1.1. *Критерії оцінки*

Метою дослідження було формування адаптивного методу навчання нейронних мереж. Було вирішено базувати даний метод на основі композиційної архітектури.

Основним критерієм оцінки ефективності методу є дискретна точність класифікації навченої нейронної мережі, що оцінює відповідність результуючого вектора класифікації із вихідним вектором, наданим у тестовому наборі. Значення точності знаходиться у діапазоні від 0 до 1 або від 0% до 100% відповідно.

Другим критерієм оцінки було обрано кількість операцій, що виконується в рамках матричного множення при обчисленні наступного шару мережі.

Третій критерій полягав у тому, наскільки розроблений програмний комплекс відповідає поставленим вимогам (кожна з вимог із високим пріоритетом, наведених у підрозділі 3.1 була реалізована).

4.1.2. *Результати експериментів*

Порівняльна характеристика точності класифікації одиночних та композиційних моделей представлена в таблицях 4.1, 4.2. В дужках представлено процентну зміну точності композиції по відношенню до одиночної моделі. Композиційні моделі формуються за принципом, описаним в підрозділі 3.3.

Таблиця 4.1

Точність класифікації одиночних та композиційних моделей без використання регуляризаційних технік (дропаут)

Одиночна	Паралельна композиція	Послідовна композиція
16-8_50_4_0.25		
0.74	0.81 (+0.07)	0.81 (+0.07)
16-8_50_4_0.75		
0.73	0.76 (+0.03)	0.76 (+0.03)
8-16-8_50_4_0.25		
0.63	0.81 (+0.18)	0.8 (+0.17)
8-16-8_50_4_0.75		
0.53	0.77 (+0.24)	0.76 (+0.23)
8-16-8_150_8_0.25		
0.99	0.99	0.99
8-16-8_150_8_0.75		
0.78	0.99 (+0.11)	0.99 (+0.11)
8-16-32-32-16-8_150_4_0.25		
0.66	0.81 (+0.15)	0.79 (+0.13)
8-16-32-32-16-8_150_4_0.75		
0.70	0.75 (+0.05)	0.77 (+0.07)
8-16-32-32-16-8_150_8_0.25		
0.99	0.99	0.99

Продовження табл. 4.1

8-16-32-32-16-8_150_8_0.75		
0.60	0.99 (+0.33)	0.99 (+0.33)

Таблиця 4.2

Точність класифікації одиночних та композиційних моделей при використанні регуляризаційних техніки (дропаут)

Одиночна	Паралельна композиція	Послідовна композиція
16-8_50_4_0.25		
0.99	0.82 (-0.17)	0.92 (-0.07)
16-8_50_4_0.75		
0.98	0.72 (-0.26)	0.75 (-0.23)
8-16-8_50_4_0.25		
0.99	0.79 (-0.2)	0.92 (-0.07)
8-16-8_50_4_0.75		
0.97	0.6 (-0.37)	0.87 (-0.1)
8-16-8_150_8_0.25		
0.99	0.99	0.99
8-16-8_150_8_0.75		
0.99	0.99	0.99
8-16-32-32-16-8_150_4_0.25		
0.98	0.79 (-0.19)	0.92 (-0.06)
8-16-32-32-16-8_150_4_0.75		
0.87	0.73 (-0.14)	0.87
8-16-32-32-16-8_150_8_0.25		
0.99	0.99	0.99
8-16-32-32-16-8_150_8_0.75		
0.99	0.99	0.99

Бачимо, що композиційні моделі дають суттєвий виграш за умови, що техніка дропауту не використовується. Середній виграш в точності як для паралельних, так і для послідовних композиційних моделей, в порівнянні з одиночними, складає 10-15%.

При використанні дропауту точність композиційних моделей суттєво падає. Для паралельних моделей втрата в точності складає в середньому 20%, для послідовних – 7%. Причиною таких результатів ми вважаємо недостатню кількість нейронів в кожній підмережі для необхідності використання дропауту.

Різниця в кількості виконаних операцій представлена в таблиці 4.3.

Таблиця 4.3

Кількість виконаних операцій при роботі нейромереж

Структура прихованих шарів	Кількість операцій одиночної мережі	Кількість операцій композиції
16-8	128	32
8-16-8	256	64
8-16-32-32-16-8	2304	576

4.2. Аналіз сформульованого критерію оцінки навченості нейронної мережі

4.2.1. *Критерії оцінки*

Предметом дослідження стала робота над критерієм оцінки навченості нейронної мережі. Було вирішено базувати критерій на основі контінуальної множини прикладів, що утворюється в рамках дискретного тестового набору даних.

Основною та єдиною метрикою було обрано ступень експресивності результатів при використанні критерію.

В якості формату представлення результатів було обрано метод візуалізації. Результати класифікацію візуалізуються на графіку, який представляє собою двовимірну площину. Область площини, представлена на графіку, відповідає квадрату зі стороною *l*, яка специфікована наданим набором даних. Візуалізована область представляє собою контінуальну множину прикладів, на якій здійснюється оцінка точності класифікації мережі. Інтенсивність точки на графіку представляє собою рівень похибки на відповідному прикладі (чим вища інтенсивність, тим вища похибка). Граничне значення похибки, яке було обрано для даної модельної задачі, становить 0.2.

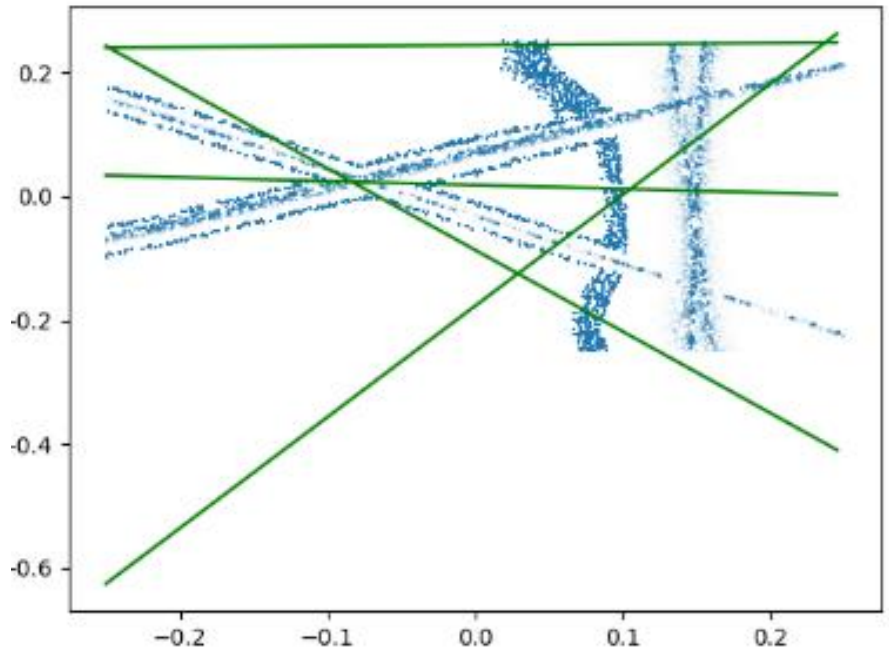
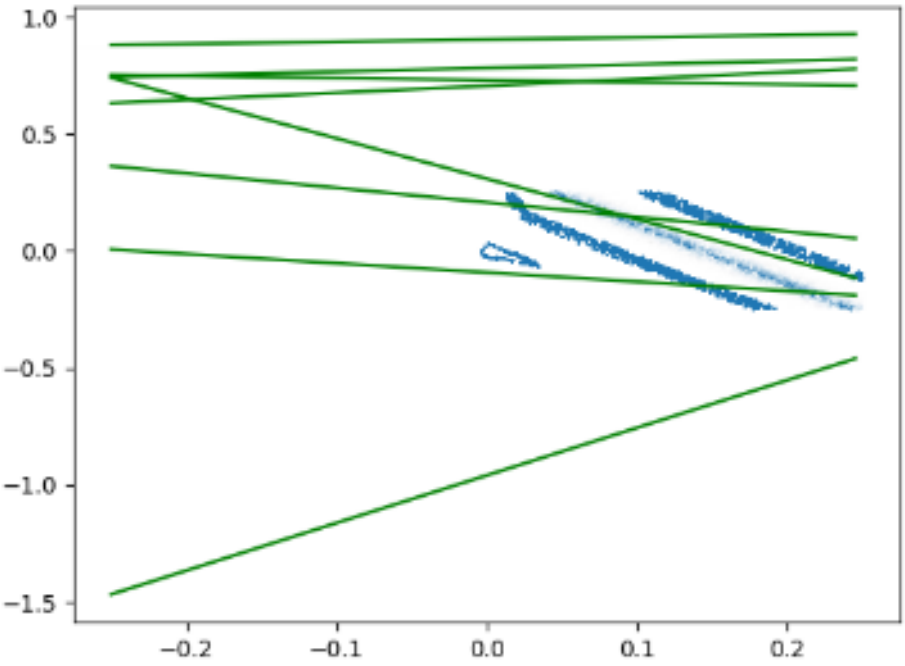
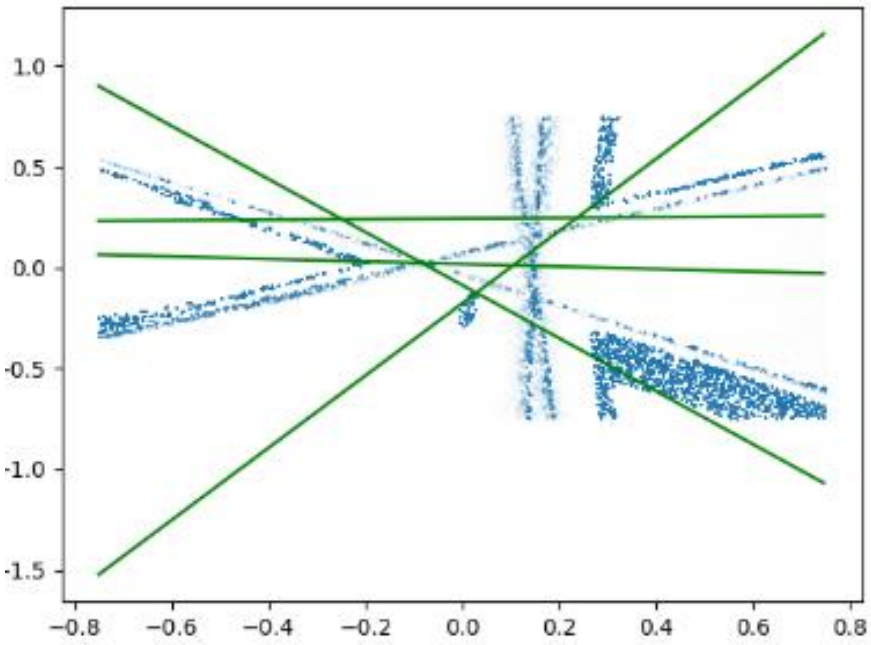


Рис. 4.1. Результати класифікації нейромережі, закодованої у *P_i_j_k* форматі: 16-8_50_4_0.25



(а)



(б)

Рис. 4.2. Результати класифікації нейромереж, закодованих у *P_i_j_k* форматі: (а) демонструє 8-16-8_50_4_0.75, (б) демонструє 8-16-32-32-16-8_150_8_0.75

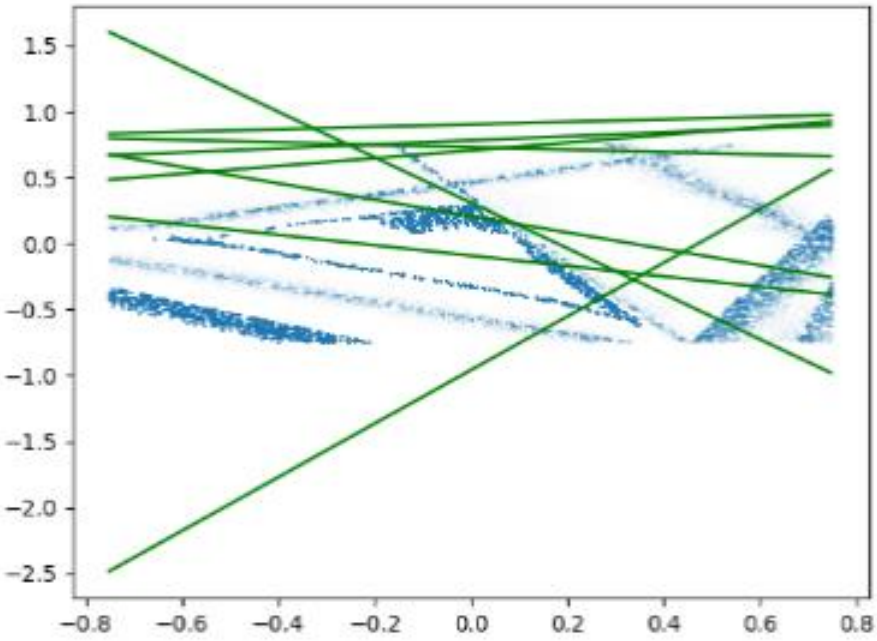
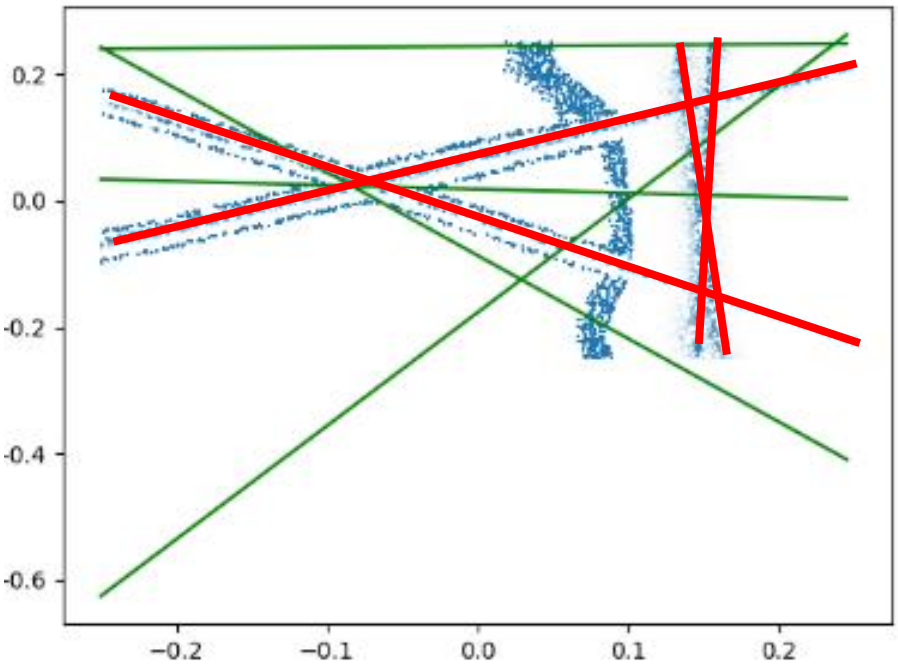


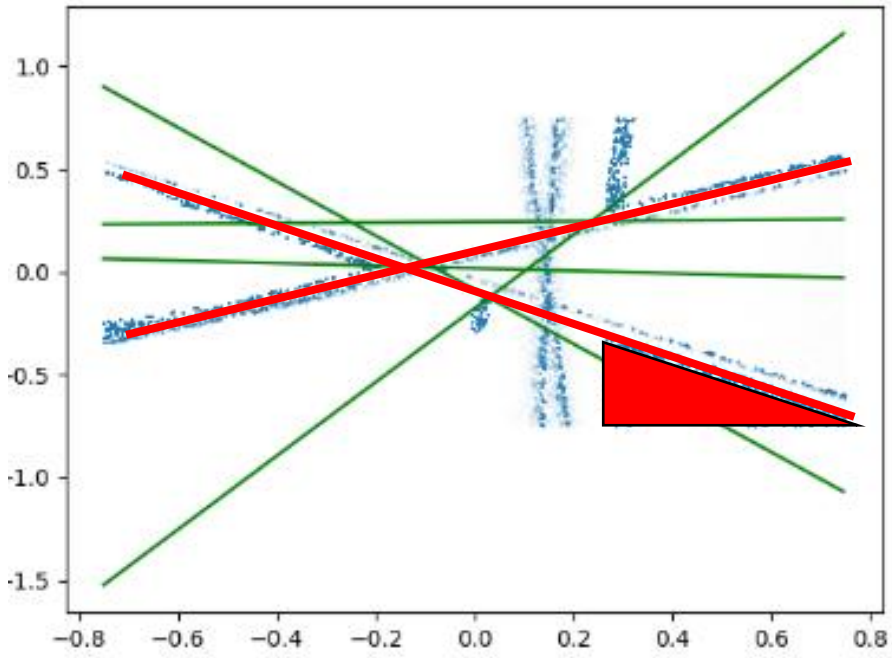
Рис. 4.3. Результати класифікації нейромережі, закодованих у *P_i_j_k* форматі: 8-16-8_150_8_0.25

В усіх чотирьох випадка мережі класифіковані як недонавчені, так як в рамках дискретного набору існують приклади, похибка класифікації яких вища за встановлену.

Важливо відмітити, що візуально можливо помітити кластери прикладів (деякі з них виділені червоним на рис. 4.4).



(а)



(б)

Рис. 4.4. Піднабори прикладів, похибка класифікації яких є нижчою за порогову: (а) демонструє 16-8_50_4_0.25, (б) демонструє 8-16-8_50_4_0.75

Бачимо, що кластери за своєю формою та положенням нагадують обмеження. На нашу думку, форма кластерів залежить від функції активації та форми обмежень, положення кластерів – від положення обмежень.

Відповідно, даний критерій дає відповідь на два питання:

- Який стан має мережа після класифікації тестового набору. Отримавши будь-який статус, окрім «навчена», мережа не може використовуватись для роботи на реальній задачі.
- Які кластери прикладів вносять найбільшу похибку у результати. Для точного визначення кластерів можливе використання додаткових кластеризаційних алгоритмів (наприклад, K-Means).

4.3. Подальший напрям дослідження

4.3.1. *Адаптивний метод навчання нейромереж*

Розроблений метод планується досліджувати в наступних напрямках:

- Використання методу для масштабніших структур нейромереж – на практиці, зазвичай, архітектури нейронних мереж включать в себе як більшу кількість шарів, так і більшу кількість нейронів в одному шарі. Для використання методу на серйозних практичних завданнях, необхідне тестування на реальних наборах даних.
- Використання методу для інших видів нейромереж – планується використання методу для наступних нейромереж: згорткові нейронні мережі, залишкові нейромережі та рекурентні нейромережі. Персептрони є зручними у використанні в рамках модельних задач, але в практичних реалізаціях зустрічаються рідше. Для тестування методу на реальних завданнях необхідні варіації методу для відповідного типу мережі.
- Адаптація отриманих модифікацій до регуляризаційних технік (таких, як дропаут).
- Формування принципів композиції нейромереж – необхідне доведене математичне формулювання, що доводить вищу ефективність методу у порівнянні із існуючими.

- Використання напрацьованої бази знань для розробки адаптивних методів навчання нейромереж на основі декомпозиції декомпозиції – з нейронних мереж можливо як створювати композиції, так і декомпозиціювати дані нейромережі з метою виділення підмереж. Дані підмережі мають класифікувати певні ознаки об'єкту, на класифікацію якого була навчена вихідна мережа. Подібна концепція була реалізована вченими Масачусетського технологічного університету [40].

4.3.2. *Критерій навченості нейронних мереж*

Для формалізації критерію необхідно дослідження наступних залежностей:

- Залежність форми кластерів від форми обмежень та функції активації (як лінійної, так і нелінійної).
- Залежність локалізації кластерів від локалізації обмежень.

Необхідно математично довести кожну з наведених залежностей для повноцінної формалізації критерію.

Окрім цього, необхідно розглянути, яким чином поводять себе кластери похибок при використанні інших типів мереж.

4.3.3. *Висновки*

Розроблений метод адаптивного навчання нейронних мереж включив в себе композиційні властивості та був оцінений за наступними метриками:

- Дискретна точність класифікації навченої нейронної мережі.
- Кількість виконаних операцій для обраних мереж.
- Відповідність розробленого продукту вимогам.

Експерименти продемонстрували, що при відсутності регуляризаційних технік, композиції мереж демонструють суттєво вищу точність класифікації (приблизно на 15%).

При використанні регуляризаційної техніки, точність класифікації композиційних мереж була суттєво нижчою для паралельних композицій (приблизно на 10%). Точність класифікації послідовних композицій була в середньому нижча на 7%.

Сформульований критерій навченості нейромереж було оцінено за результатами класифікації. На базі отриманої оцінки були виділені припущення з приводу характеристичних властивостей кластерів похибок. Експерименти показали, що, не дивлячись на високі значення точності, розглянуті мережі отримали статус недонавчених.

Були сформовані подальші шляхи дослідження, серед яких:

- Використання методу для масштабніших структур нейромереж.
- Використання методу для інших видів нейромереж.
- Адаптація модифікованого методу до регуляризаційних технік.
- Формування принципів композиції нейромереж.
- Використання напрацьованої бази знань для розробки методів декомпозиції нейромереж.
- Залежність форми кластерів від форми обмежень та функції активації (як лінійної, так і нелінійної).
- Залежність локалізації кластерів від локалізації обмежень.

ВИСНОВКИ

Нейронні мережі стали одним з найпопулярніших інструментів для розробки автоматизованих рішень. Ці моделі виявились особливо корисними для вирішення задач класифікації. Тим не менше, сучасні підходи до формування мережевої структури та навчання забирають багато часу. Перенавчання нейронних мереж, якщо це необхідно, відбувається на всіх прикладах навчального набору даних. Даний підхід робить неможливим використання результатів навчання в рамках попередньої ітерації.

Ще однією проблемою використання нейронних мереж є ситуація, у якій разом із підвищеною точністю класифікації зростає похибка при активації окремих нейронів всередині глибоких шарів. Даний факт означає, що, надаючи нейронній мережі відповідні приклади, активація певних нейронів у глибоких шарах мережі може привносити суттєві похибки в кінцеві обчислення. Надання відповідних прикладів нейронній мережі називають «змагальною атакою», метою якої є погіршення результатів класифікації.

Для контролю якості нейронних мереж, використовують два підходи: емпіричні метрики (точність, F1-міра), та методи верифікації (останні отримали суттєвий розвиток декілька років тому). Тим не менш, жоден з даних підходів не надає повного опису якості класифікації, яку виконує нейронна мережа. Емпіричні метрики надають лише оцінку дискретної множини прикладів та не можуть бути використані для опису всієї області значень множини прикладів. Методи верифікації надають можливість оцінки всієї контінуальної множини прикладів, але не дозволяють оцінити, яка саме підмножина прикладів викликає суттєву похибку у результатах класифікації.

В даній дисертації було запропоновано адаптивний метод навчання нейронних мереж на основі композиційної архітектури та критерій навченості нейронних мереж.

Запропонований метод адаптивного навчання було протестовано на двох моделях: паралельній та послідовній. Було описано та продемонстровано їх переваги перед одинарними моделями. Навчені мережі тестувались на відповідній змодельованій задачі, яка нагадує гіперплощину з множинними обмеженнями (експерименти, представлені в статті, використовували двовимірний простір). Були проведені два етапи експериментів: без використання технології відсіву та разом з нею.

Згідно з отриманими цифрами, середній приріст точності при використанні адаптивного методу навчання становить 10-15% порівняно з одинарними мережами (без використання регуляризаційної техніки дропауту). Це пов’язано з тим, що кожна підмережа „фокусується” на об’єктах, схожих між собою. При використанні регуляризаційної техніки дропауту окремі загальні мережі забезпечують вищу точність, ніж відповідні мережі, що базуються на композиційній архітектурі. Швидше за все, цей ефект виникає через невелику кількість нейронів у мережах. Наведена порівняльна характеристика кількості обчислювальних операцій, що виконується при навчанні нейронної мережі суцільної архітектури та при навчанні нейронної мережі композиційної архітектури (використовуючи адаптивний метод навчання). Продемонстровано формальні умови, за яких кількість обчислювальних операцій при використанні методу адаптивного навчання буде завжди меншою, ніж кількість обчислювальних операцій, що виконуються при використанні нейронної мережі із суцільною архітектурою.

Окрім адаптивного методу навчання, дисертація пропонує критерій оцінки навченості нейронної мережі, який розглядає похибку класифікації як контінуальну множину. Аналогічно, множина прикладів тестового набору даних розглядається як контінуальна.

Критерій формується за допомогою опису станів навченості нейронної мережі, кожен з яких специфікується відповідністю множини похибок до області визначення функції, яка описує контінуальну множину прикладів тестового набору:

- Ненавчена.
- Недонавчена.
- Навчена.
- Перенавчена.

Описаний критерій дозволяє не тільки ідентифікувати, що нейронна мережа не є достатньо навченою на відповідній множині прикладів, але й оцінити, які саме приклади (або піднабори прикладів викликають похибку). Це означає, що критерій може використовуватись як в якості незалежної оцінки, так і комбінації із іншими методами.

В дисертації наводиться опис моделі, на якій проводились експерименти, що представляє собою гіперплощину із рядом обмежень. В якості класів використовуються набори операторів, кожен з яких обумовлює певну область, на які гіперплощина ділиться поставленими обмеженнями (експерименти, наведені у дисертації, використовують двовимірний простір). Отримані результаті були отримані при використанні трьох архітектур персептронів на відповідних наборах даних. Візуалізація демонструє, що на результати класифікації нейронних мереж мають вплив певні піднабори прикладів. Більше того, після перенавчання моделей нейромереж, піднабори прикладів зберігають свій топологічний вигляд на візуалізованих графіках не дивлячись на існуючий вірогідносний фактор, який обумовлює процес навчання. Фільтрація результатів класифікації за граничною величиною похибки демонструє, що у більшості випадків піднабори певним чином відповідають обмеженням, які накладаються на вихідну задачу класифікації.

Додатково, в дисертації наводиться опис програмної реалізації та проектування програмного комплексу, який використовувався для тестування наведених методів та концепцій.

Розроблений програмний комплекс включив в себе 5 модулів, серед яких:

- Модуль генерації наборів даних.
- Модуль передобробки наборів даних.
- Модуль генерації конфігураційних файлів.
- Модуль навчання моделей.
- Модуль візуалізації та збереження результатів.

В дисертації наведено опис кожного з модулів та список вимог, які були необхідними для виконання, щоб гарантувати успішну розробку та тестування запропонованих методів. Запропоновано опис технологій, мов та бібліотек, які були використані при розробці, серед яких:

- Maxima v. 5.42.
- Tensorflow v. 2.0 – тензорна бібліотека машинного навчання.
- Keras v. 2.0 – окремий інтерфейс Tensorflow для роботі із моделями нейронних мереж.
- Matplotlib – бібліотека для побудови та візуалізації графіків.
- Numpy – бібліотека для мови програмування Python, що додає підтримку великих багатовимірних масивів та матриць, а також велику колекцію математичних функцій високого рівня для роботи з цими масивами.
- Scikit-Learn – безкоштовна програмна бібліотека машинного навчання для мови програмування Python. Він має різні алгоритми класифікації, регресії та кластеризації, включаючи машини векторної підтримки, тощо.

Одним з подальших напрямків дослідження є перехід від персептронів до згорткових та рекурентних мереж та залучення нових базових композицій. Окремим предметом дослідження є оцінка похибки

континууму, отриманої в результаті класифікації. На думку авторів, обмеження (та форма) піднаборів прикладів, на яких рівень похибки є вищим за допустимим, залежать від структури мережі та функцій активації, що використовуються як в прихованих шарах, так і у вихідному шарі. Окрім цього, планується фокусувати увагу на наступних аспектах дослідження:

- Використання методу для масштабніших структур нейромереж.
- Використання методу для інших видів нейромереж.
- Адаптація модифікованого методу до регуляризаційних технік.
- Формування принципів композиції нейромереж.
- Використання напрацьованої бази знань для розробки методів декомпозиції нейромереж.
- Залежність форми кластерів від форми обмежень та функції активації (як лінійної, так і нелінійної).
- Залежність локалізації кластерів від локалізації обмежень.

СПИСОК ЛІТЕРАТУРИ

1. Thomas Ritter, Carsten Lund Pedersen. Digitization capability and the digitalization of business models in business-to-business firms: Past, present, and future. In: Industrial Marketing Management, Vol. 86, pp. 180-190., April 2020.
2. Kaur, Gurmeet & Bajaj, Karan. News Classification using Neural Networks. In: Communications on Applied Electronics, Vol. 5 (1), DOI:10.5120/cae2016652224, pp. 42-45, 2016.
3. E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui and J. Long. A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture. In: IEEE Access, vol. 6, DOI: 10.1109/ACCESS.2018.2855437, pp. 39501-39514, 2018.
4. Toviah Moldwin, Idan Segev. Perceptron Learning and Classification in a Modeled Cortical Pyramidal Cell. In: Frontiers in Computational Neuroscience, 24 April 2020.
5. Khan, A., Sohail, A., Zahoora, U. et al. A survey of the recent architectures of deep convolutional neural networks. In: Artificial Intelligence Review 53, 5455–5516, 2020.
6. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. arXiv preprint arXiv:1512.03385v1. 2015.
7. Pengfei Liu, Xipeng Qiu, Xuanjing Huang. Recurrent Neural Network for Text Classification with Multi-Task Learning. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence. arXiv preprint arXiv:1512.03385v1. 2015.
8. Andreas Venzke, Senior Member. Verification of Neural Network Behaviour: Formal Guarantees for Power System Applications. arXiv preprint arXiv: 1910.01624v4. 2020.

9. Changliu Liu, Tomer Arnon, Christopher Lazarus, Clark Barrett, Mykel J. Kochenderfer. Algorithms for Verifying Deep Neural Networks. arXiv preprint arXiv:1903.06758v2. 2020.
10. Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., Jun. 2014.Generative Adversarial Networks. arXiv:1406.2661 [cs, stat]ArXiv: 1406.2661.
11. Hinton, Geoffrey; Sejnowski, Terrence (1999). Unsupervised Learning: Foundations of Neural Computation. MIT Press. ISBN 978-0262581684.
12. IJ Goodfellow, J. Shlens, and C. Szegedy, «Пояснення і використання змагальних прикладів», Міжнародна конференція по навчальним уявленням (ICLR, 2015), 2015.
13. Zehao Douy, Stanley J. Osher, Bao Wangz. Mathematical Analysis of Adversarial Attacks. arXiv preprint arXiv:1811.06492v2. 2018
14. Andreas Venzke, Senior Member. Verification of Neural Network Behaviour: Formal Guarantees for Power System Applications. arXiv preprint arXiv: 1910.01624v4. 2020.
15. Changliu Liu, Tomer Arnon, Christopher Lazarus, Clark Barrett, Mykel J. Kochenderfer. Algorithms for Verifying Deep Neural Networks. arXiv preprint arXiv:1903.06758v2. 2020.
16. Veldhoen N, Hobbs J, Ikonomou G, Hii M, Lesperance M, et al. (2016) Implementation of Novel Design Features for qPCR-Based eDNA Assessment. PLOS ONE 11(11): e0164907. https://doi.org/10.1371/journal.pone.0164907
17. David Saxton, Edward Grefenstette, Felix Hill, andPushmeet Kohli. 2019. Analysing mathematical rea-soning abilities of neural models.arXiv preprintarXiv:1904.01557.
18. Lake, B. M.; and Baroni, M. 2017.Generalizationwithout systematicity: On the compositional skills ofsequence-to-sequence recurrent networks.arXiv preprint arXiv:1711.00350.

19. J. Loula, M. Baroni, and B. M. Lake. Rearranging the Familiar: Testing CompositionalGeneralization in Recurrent Networks. jul 2018. URLhttp://arxiv.org/abs/1807.07545.
20. R. Dessì and M. Baroni. CNNs found to jump around more skillfully than RNNs: Compositionalgeneralization in seq2seq convolutional networks.arXiv:1905.08527 [cs], May 2019.
21. Liska, A., Kruszewski, G., and Baroni, M. (2018). Memorize or generalize? Searching for a compo-sitional RNN in a haystack. InProceedings of AEGAP (FAIM Joint Workshop on Architecturesand Evaluation for Generality, Autonomy and Progress in AI).
22. Bowman, S. R., Manning, C. D., and Potts, C. (2015). Tree-structured composition in neural net-works without tree-structured architectures. InProceedings of the 2015th International Conferenceon Cognitive Computation: Integrating Neural and Symbolic Approaches, pages 37–42.
23. Mul, M. and Zuidema, W. (2019). Siamese recurrent networks learn first-order logic reasoning andexhibit zero-shot compositional generalization.CoRR, abs/1906.00180.
24. Tran, K., Bisazza, A., and Monz, C. (2018). The importance of being recurrent for modelinghierarchical structure. InProceedings of the 2018 Conference on Empirical Methods in NaturalLanguage Processing (EMNLP), pages 4731–4736.
25. Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. The compositionality of neuralnetworks: integrating symbolism and connectionism.arXiv preprint arXiv:1908.08351, 2019.
26. Y. Lu and F. Salem, “Simplified gating in long short-term memory (LSTM) recurrent neural networks,” arXiv:1701.03441
27. H. Kameoka, K. Tanaka, T. Kaneko, and N. Hojo, “ConvS2S-VC: Fully convolutional sequence-to-sequence voice conver-sion,”arXiv preprint arXiv:1811.01609, 2018.

28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones,L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Atten-tion is all you need.arXiv:1706.03762, 2017.
29. A.D. Garcez and L.C. Lamb. 2020. Neurosymbolc AI: The 3rd Wave.arXiv:2012.05876v2(2020).
30. L. Serafini and A.S. d'Avila Garcez. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. CoRR, abs/1606.04422, 2016.
31. G. Marra, F. Giannini, M. Diligenti, and M. Gori. Lyrics: A general interface layer to integrate logic inference and deep learning. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 283–298. Springer, 2019.
32. E. van Krieken, E. Acar, and F. van Harmelen. Analyzing Differentiable Fuzzy Implications. In Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, pages 893–903, 2020.
33. Sotirov, Sotir, Sotirova, Evdokia, Atanassova, et al. A Hybrid Approach for Modular Neural Network Design Using Intercriteria Analysis and Intuitionistic Fuzzy Logic. Complexity. 2018. 1-11. 10.1155/2018/3927951.
34. K. Chen. Deep and Modular Neural Networks. Handbook of Computational Intelligence. 2015.
35. Katz, G., Barrett, C., Dill, D., Julian, K., Kochenderfer, M.: Reluplex: An EfficientSMT Solver for Verifying Deep Neural Networks. arXiv:1702.01135 [cs] (May 2017).
36. Henriksen P, Lomuscio A, 2020, Efficient Neural Network Verification via Adaptive Refinement and Adversarial Search, European Conference on Artificial Intelligence (ECAI20).
37. Gagandeep Singh, Timon Gehr, Markus Püschel, Martin T Vechev. An abstract domain for certifying neural networks. In: Proceedings of the

ACM on Programming Languages, No. 41, DOI: 10.1145/3290354. 2019.

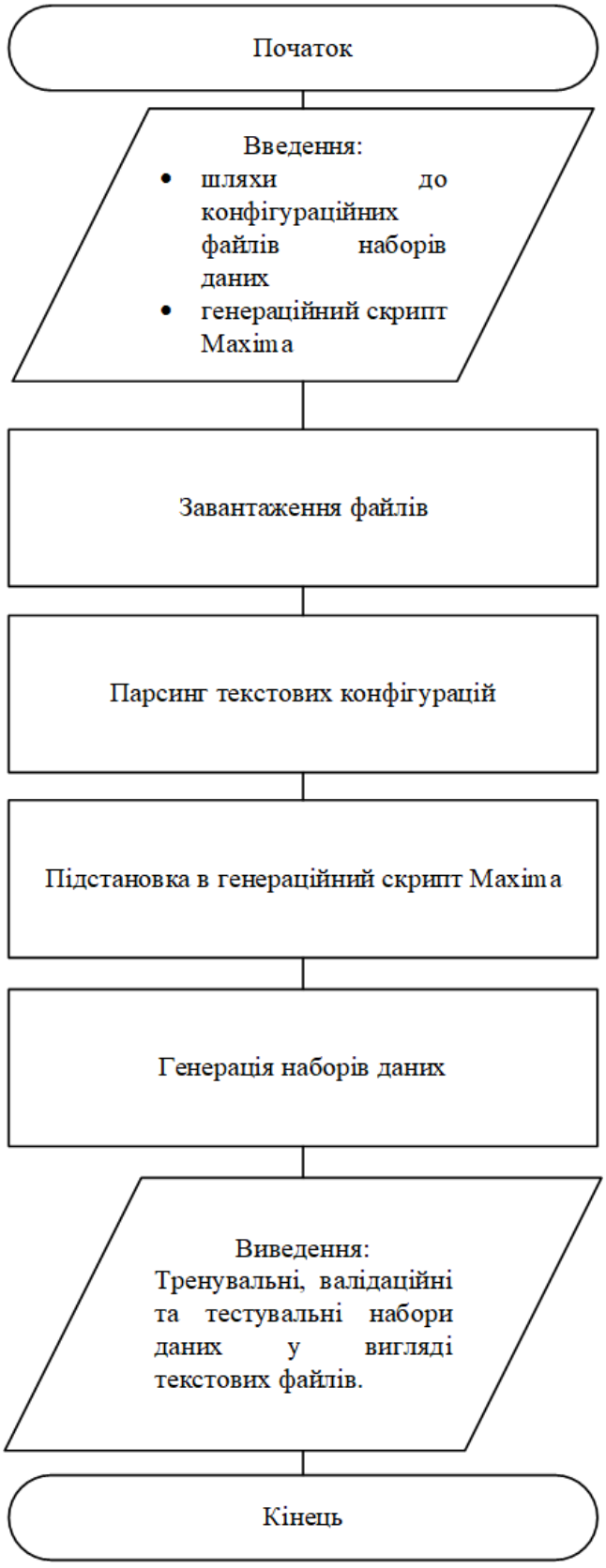
38. Hoang-Dung Tran et al. Star-Based Reachability Analysis of Deep Neural Networks. In: International Symposium on Formal Methods, FM 2019: Formal Methods – The Next 30 Years. DOI: 10.1007/978-3-030-30942-8_39. 2019.
39. T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri and M. Vechev, "AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation," 2018 IEEE Symposium on Security and Privacy (SP), 2018, pp. 3-18, doi: 10.1109/SP.2018.00058.
40. Frankle, J. and Carbin, M. The lottery ticket hy-pothesis: Finding sparse, trainable neural networks.arXiv:1803.03635, 2018.

ДОДАТКИ

Додаток 1

Копії графічних матеріалів

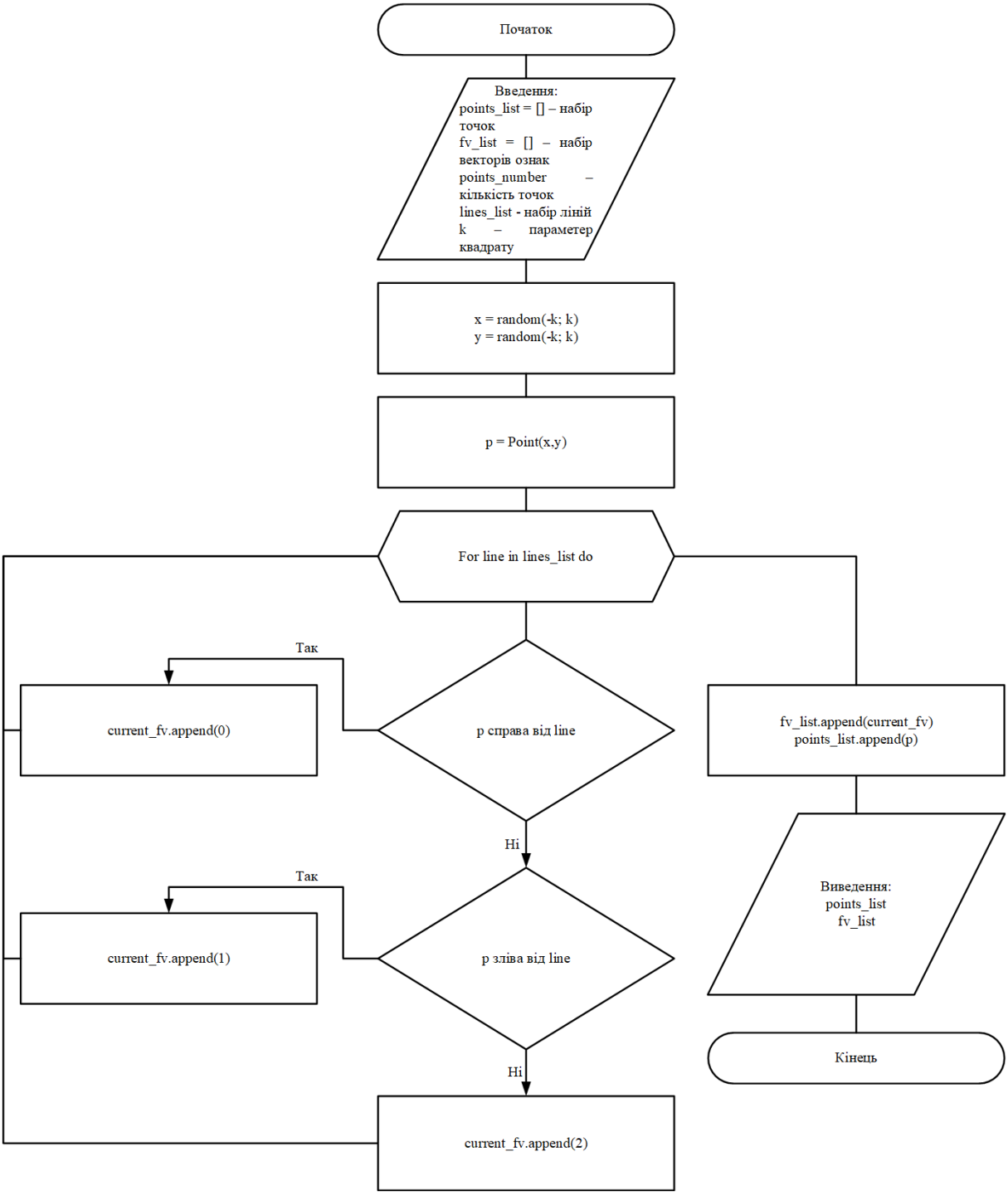
Алгоритм генерації наборів даних



Михайло Іващенко

КП-91 мн

Алгоритм генерація набору даних



Михайло Іващенко

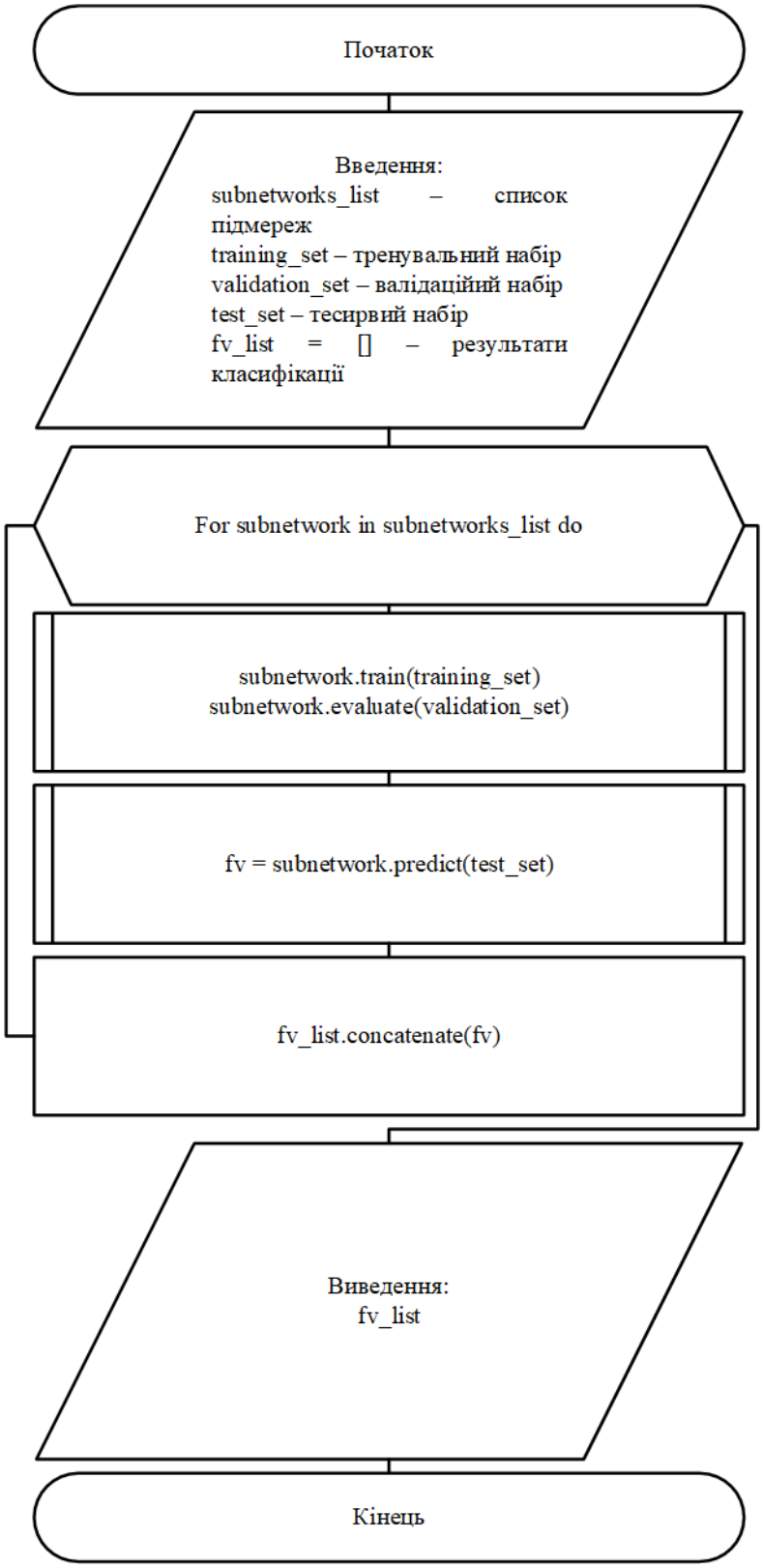
КП-91 мн

Алгоритм компіляції моделей нейронних мереж

Михайло Іващенко

КП-91 мн

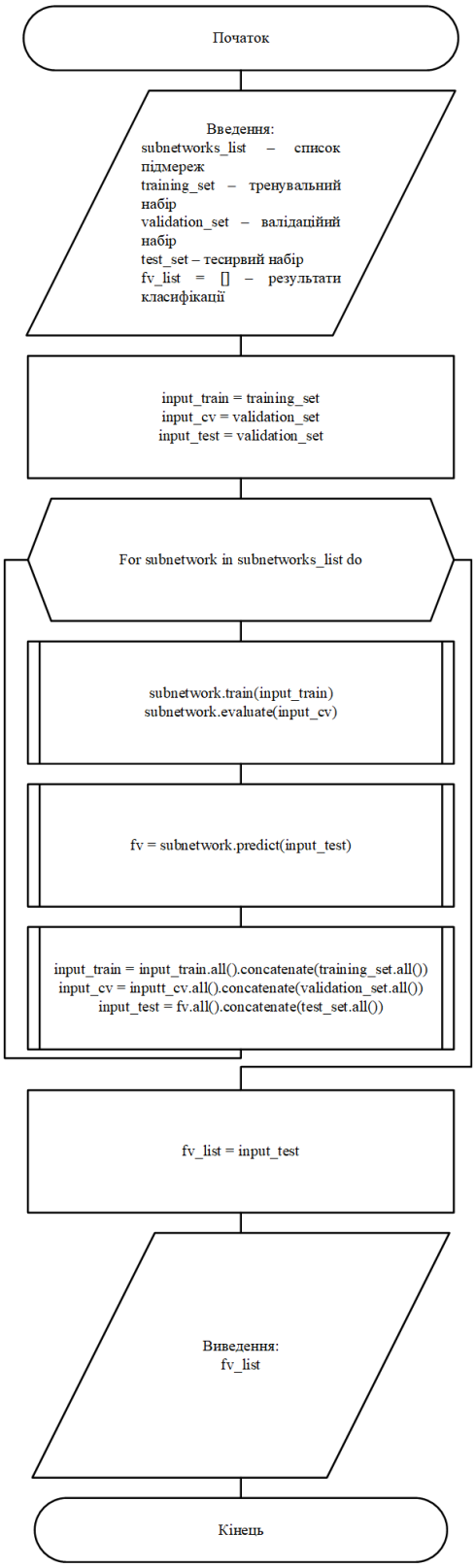
Вирішення задачі класифікації методом адаптивного навчання паралельної композиційної архітектури



Михайло Іващенко

КП-91 мн

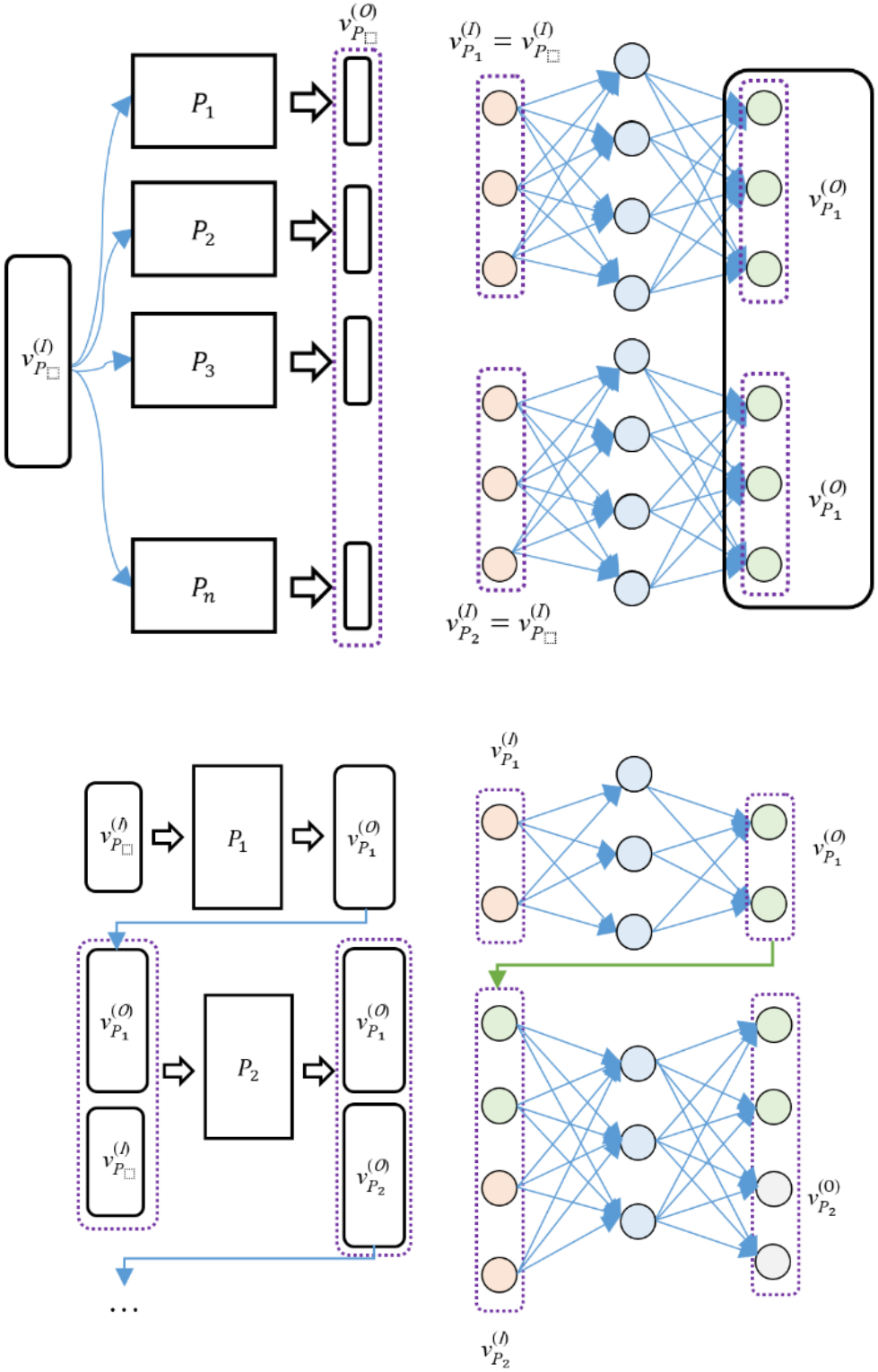
Вирішення задачі класифікації методом адаптивного навчання послідовної композиційної архітектури



Михайло Іващенко

КП-91 мн

Графічна схема адаптивних методів навчання, заснованих на паралельні та послідовній композиційних архітектурах



Михайло Іващенко

КП-91 мн

Додаток 2

Фрагменти тексту програми

Лістинг 1. Фрагмент коду, який реалізовує модель генерації синтетичних даних

```
load ("eigen")$

defstruct (Vector (vectorStart, vectorEnd))$
defstruct (Line (directionVec, normalEnd))$

calcDeterminant(testLine) := block(
    [],
    offset           :           testLine@directionVec@vectorEnd          -
testLine@directionVec@vectorStart,
    offset_left : copylist (offset),
    offset_left[1] : offset_left[1] * (-1),
    offset_right : copylist(offset),
    offset_right[2] : offset_right[2] * (-1),
    m1 : matrix(offset, offset_left),
    if determinant(m1) > 0
        then
            return(offset_left + testLine@directionVec@vectorStart)
        else
            return(offset_right +testLine@directionVec@vectorStart)
) $

calcNormals(testLines) :=(
    map(lambda([line], line@normalEnd : calcDeterminant(line)), testLines)
) $

composeParametric(line)  :=  parametric(line@directionVec@vectorStart[X]  +
line@directionVec@vectorEnd[X] * t,
                                        line@directionVec@vectorStart[Y]  +
line@directionVec@vectorEnd[Y] * t,
                                        t,                LEFT_LINE_THRESH,
RIGHT_LINE_THRESH)$

analyzePosition(point, line) := block (
    [],
    dif : inprod((point - line@directionVec@vectorStart), (line@normalEnd -
line@directionVec@vectorStart)),
    if abs(dif) >= e
        then
            if sign(dif) = pos
                then return (LEFT)
                else return (RIGHT)
        else return (ON_THE_LINE)
) $

composeArea(featureVector) := (
    areaClasses : append(areaClasses, [[featureVector, concat(AREA_KEYWORD,
string(length(areaClasses)))]])
 ) $

composeFeature(point, testLines) := block(
    [],
    feature : "",
    disp(testLines),
    map(lambda([line],  feature  :  concat(feature,  analyzePosition(point,
line))), testLines),
    return(feature)
) $

checkPresence(currentFeature) := (
    if assoc(currentFeature[FEATURE_VECTOR_ID], areaClasses) = false
```

```
        then          (basicPoints          :          append(basicPoints,
[currentFeature[POINT_ID]]),
                  composeArea(currentFeature[FEATURE_VECTOR_ID]),
                  basicFeature            :            append(basicFeature,
[[append(currentFeature[POINT_ID],
[currentFeature[FEATURE_VECTOR_ID]])]]),
                  initPoints              :              append(initPoints,
[currentFeature[POINT_ID]]),
                  testFeature             :             append(testFeature,
[[append(currentFeature[POINT_ID],
[currentFeature[FEATURE_VECTOR_ID]])]]))
        else (initPoints : append(initPoints, [currentFeature[POINT_ID]]),
                 testFeature              :             append(testFeature,
[[append(currentFeature[POINT_ID],
[currentFeature[FEATURE_VECTOR_ID]])]]))
) $

checkIfAdjacent(point, lines) := (
    count : 0,
    for i  : 1 thru length(lines) do(
        dif    :    inprod((point    -    lines[i]@directionVec@vectorStart),
(lines[i]@normalEnd - lines[i]@directionVec@vectorStart)),
        if abs(dif) < e or abs(dif) > e1
            then (count : count + 1)
    ) ,
    if count = length(lines) then (pointsList : delete(point, pointsList))
)$

filterAdjacentToLine(testLines) := (
    map(lambda([point], checkIfAdjacent(point, testLines)), pointsList)
)$

tracePoints(testLines, pointsNumber, parameters) := block (
    set_random_state (true),
    pointsList : [],
    if parameters[PREDEFINED_FLAG] = 1
       then (pointsList : read_nested_list("predefined.txt", semicolon)),
     if parameters[RANDOM_FLAG] = 1
       then            (pointsList            :            append(pointsList,
create_list([random(RIGHT_THRESH + abs(LEFT_THRESH)) + LEFT_THRESH,
                random(RIGHT_THRESH + abs(LEFT_THRESH)) + LEFT_THRESH], i,
1, pointsNumber))),
     if parameters[SAVE_PRREDEFINED_FLAG] = 1
          then (predefined : append(predefined, pointsList)),
    /*filterAdjacentToLine(testLines),*/
    map(lambda([point],     checkPresence([point,     composeFeature(point,
testLines)])), pointsList)
) $
```

Лістинг 2. Фрагмент класу, який зберігає та маніпулює моделями, та їх композиціями

```
class BasicContainer(object):

    def __init__(self, configPaths):
        self.configData = []
        parser = DefaultParser()

        for configData in configPaths:
            config = open(configData, "r")

            parameters = config.readlines()
            stringDict = "{ " + ''.join(parameters).replace("\n\"", ", \"",
len(parameters) - 1) + " }"

            paramDict = ast.literal_eval(stringDict)

            self.configData.append(paramDict)

        self.modelsList = []
        self.clusters = []

    #TODO: define learning algorithm order => how to create clusters, how
to init the models, how to launch models' learning
    def initModels(self, initIDs = ALL_CONFIGS):
        if(initIDs == ALL_CONFIGS):
            initConfigs = self.configData
        else:
            initConfigs = [self.configData[id] for id in initIDs]

        # TODO: add init status
        for config in initConfigs:
            currentModel = None

            if config[configKeywords[NN_TYPE_ID]] == PERCEPTRON_ID:
                currentModel = Perceptron(configData = config,
learning_rate_decay_flag=config[configKeywords[LEARNING_RATE_DECAY_FLAG_ID]
], \
                learning_rate = config[configKeywords[LEARNING_RATE_ID]],
output_activation=config[configKeywords[OUTPUT_ACTIVATION_ID]], \
                optimizer = config[configKeywords[OPTIMIZER_ID]],
batch_size = config[configKeywords[MINI_BATCH_SIZE_ID]],\
                epochs_num = config[configKeywords[EPOCHS_ID]],
hidden_layers = config[configKeywords[HIDDEN_LAYERS_STRUCT_ID]])

            self.modelsList.append(currentModel)


        return self.modelsList

    def __getitem__(self, id):
        return self.modelsList[id]

    # TODO: add init status check
    def trainModels(self, datascope = None, assignmentDict = None):
        if assignmentDict is None:
            for dataset in datascope.getDatasets():
                X_train, y_train, X_cv, y_cv, X_test, y_test =
dataset.unload()

                for currentModel in self.modelsList:
```

```
                    currentModel.initSession(y_train.shape[1],
X_train.shape[1], y_train.shape[1])
                    currentModel.train(X_train, y_train, X_cv, y_cv,
X_test, y_test)
        else:
            for modelID in assignmentDict.keys():
                X_train, y_train =
datascope[assignmentDict[modelID][TRAIN_DS_ID]].unloadTrain(SHUFFLE_OFF)
                X_cv, y_cv =
datascope[assignmentDict[modelID][CV_DS_ID]].unloadCV(SHUFFLE_OFF)
                X_test, y_test =
datascope[assignmentDict[modelID][TEST_DS_ID]].unloadTest(SHUFFLE_OFF)

                self.modelsList[modelID].initSession(y_train.shape[1],
X_train.shape[1], y_train.shape[1])
                self.modelsList[modelID].train(X_train, y_train, X_cv,
y_cv, X_test, y_test,
datascope[assignmentDict[modelID][TRAIN_DS_ID]].get_name())

    def predictModels(self, datascope = None, assignmentDict = None,
shuffleFlags = SHUFFLE_OFF):
        result = {}

        for modelID in assignmentDict.keys():
            result[modelID] =
self.modelsList[modelID].predict(datascope[assignmentDict[modelID]].unloadT
est(shuffleFlags)[X_ID])

        return result

    def createCluster(self, modelsIDs = ALL_MODELS, learningMode =
SEPARATE_LEARNING, predictingMode = SEPARATE_PREDICTION, models_masks =
[]):
        clusterModels = []
        if(modelsIDs == ALL_MODELS):
            clusterModels = self.modelsList
        else:
            clusterModels = [self.modelsList[modelID] for modelID in
modelsIDs]

        self.clusters.append(BasicCluster(clusterModels, learningMode,
predictingMode, models_masks))

    def trainClusters(self, datascope = None, assignmentDict = None,
shuffleFlags = SHUFFLE_OFF):

        for clusterID in assignmentDict.keys():
            datasetsIDs = list(dict.fromkeys([dsID for dsList in
assignmentDict[clusterID].values() for dsID in dsList]))

            idMap = {}

            for dsID in datasetsIDs:
                idMap[dsID] = datascope[dsID]

            self.clusters[clusterID].train(self.clusters[clusterID].get(),
idMap, assignmentDict[clusterID], shuffleFlags)

    def predictClusters(self, datascope = None, mergeOutputPredictionDict =
None, shuffleFlags = SHUFFLE_OFF):
        result = {}
```

```
        for clusterID in mergeOutputPredictionDict.keys():
            result[clusterID] =
self.clusters[clusterID].predict(self.clusters[clusterID].get(),
[datascope[i] for i in mergeOutputPredictionDict[clusterID]], shuffleFlags)

        return result
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

АДАПТИВНІ МЕТОДИ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ ЗА НАЯВНОСТІ ОБМЕЖЕНИХ ОБЧИСЛЮВАЛЬНИХ РЕСУРСІВ

Виконав: Іващенко Михайло Вікторович

Науковий керівник: Ст. викладач кафедри ПЗКС, к.т.н., Люшенко Л.А.

Київ 2021

АКТУАЛЬНІСТЬ

- Кількість часових ресурсів, які витрачаються на програмне навчання нейромереж зростає експоненційно в залежності від рівня важкості задачі.
- Глибина мереж та розмір шарів продовжують зростати із процесом розвитку глибинного навчання.
- При роботі із системами, обчислювальні ресурси яких суттєво обмежені, використання об’ємних нейронних мереж та організація навчання в режимі реального часу неможливі.

АКТУАЛЬНІСТЬ (ПРОДОВЖЕННЯ)

- Методи верифікації не надають характеристичної оцінки, якщо нейромережа класифікується як непридатна до використання.
- Основою циклу навчання нейромереж є навчання кожного наступного екземпляру з нуля.
- **Наслідки**:
  - Розміщення об’ємних мереж в системах з обмеженими обчислювальними ресурсами є неможливим.
  - Навчання в режимі реального часу в системах з обмеженими обчислювальними ресурсами є неможливим.
  - Використання результатів навчання нейромережі в рамках інших задач є проблематичним.

ПОСТАНОВКА ЗАДАЧІ

Мета дослідження полягає у підвищенні ефективності процесу навчання нейронних мереж в рамках обмежених обчислювальних ресурсів.

Завдання:

1. Дослідити існуючі методи навчання та верифікації нейромереж.
2. Виявити недоліки використання даних методів при роботі із системами обчислювальні ресурси яких є обмеженими.
3. Розробити методи адаптивного навчання та критерій оцінки навченості нейромереж для зменшення витрат ресурсів та часу, що використовуються в рамках циклу навчання.
4. Розробити синтетичну модель генерації даних для тестування сформованих наукових гіпотез.
5. Розробити програмне забезпечення для навчання моделей нейронних мереж та генерації синтетичних даних.
6. Реалізувати і протестувати сформульовані методи та критерій на базі розробленого програмного забезпечення.

АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ НАВЧАННЯ

- Основні категорії:
  - Навчання із вчителем.
  - Навчання без вчителя.
- Цикл навчання:
  1. Визначення типу навчальних прикладів.
  2. Збір прикладів та формулювання навчального набору.
  3. Формулювання навчальних ознак.
  4. Вибір алгоритму навчання.
  5. Завершення побудови моделі. У випадку незадовільного результату – Крок №1.

НЕДОЛІКИ ІСНУЮЧИХ МЕТОДІВ НАВЧАННЯ

- Відсутність можливості адаптивного навчання.
- Необхідність виконання великої кількість обчислювальних операцій.
- Неможливість навчання в режимі реального часу.
- **Спроби вирішити проблему**:
  - Пошук нейромереж меншого об’єму (Frankle et al., 2020).
  - Апаратні реалізації (FPGA).

АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ВЕРИФІКАЦІЇ

- Основні категорії:
  - SMT-методи.
  - Алгоритми адаптивного уточнення та змагального пошуку.
  - Метод абстрактного домену та похідні від нього.
- Перевага:
  - Надання позитивної/негативної відповіді на питання придатності мережі для вирішення відповідної задачі.
- Недоліки:
  - Відсутність причинної характеристики наданої відпвоіді, якщо така була негативною.

ПАРАМЕТРИ СИСТЕМ З ОБМЕЖЕНИМИ РЕСУРСАМИ

- Пам’ять – до 4 Гб.
- Швидкість CPU – до 1.5 GHz.
- Віртуальна пам’ять – до 1 Гб.
- Живлення – до 2.5 А.

ГІПОТЕЗИ

- Метод навчання, розроблений на базі композиційної архітектури дозволить як досягти адаптивності, так і зберегти обчислювальні ресурси.
- Критерій, використання якого дозволяє виконати аналіз результатів розпізнавання та локалізувати піднабори прикладів, на яких нейронна мережа не демонструє необхідної точності, дозволить підвищити якість кожної наступної ітерації навчання.

ЗАПРОПОНОВАНІ МЕТОД АДАПТИВНОГО НАВЧАННЯ ТА КРИТЕРІЙ ОЦІНКИ НАВЧЕНОСТІ НЕЙРОМЕРЕЖ

СПРОЩЕНЕ ФОРМУЛЮВАННЯ МОДЕЛІ

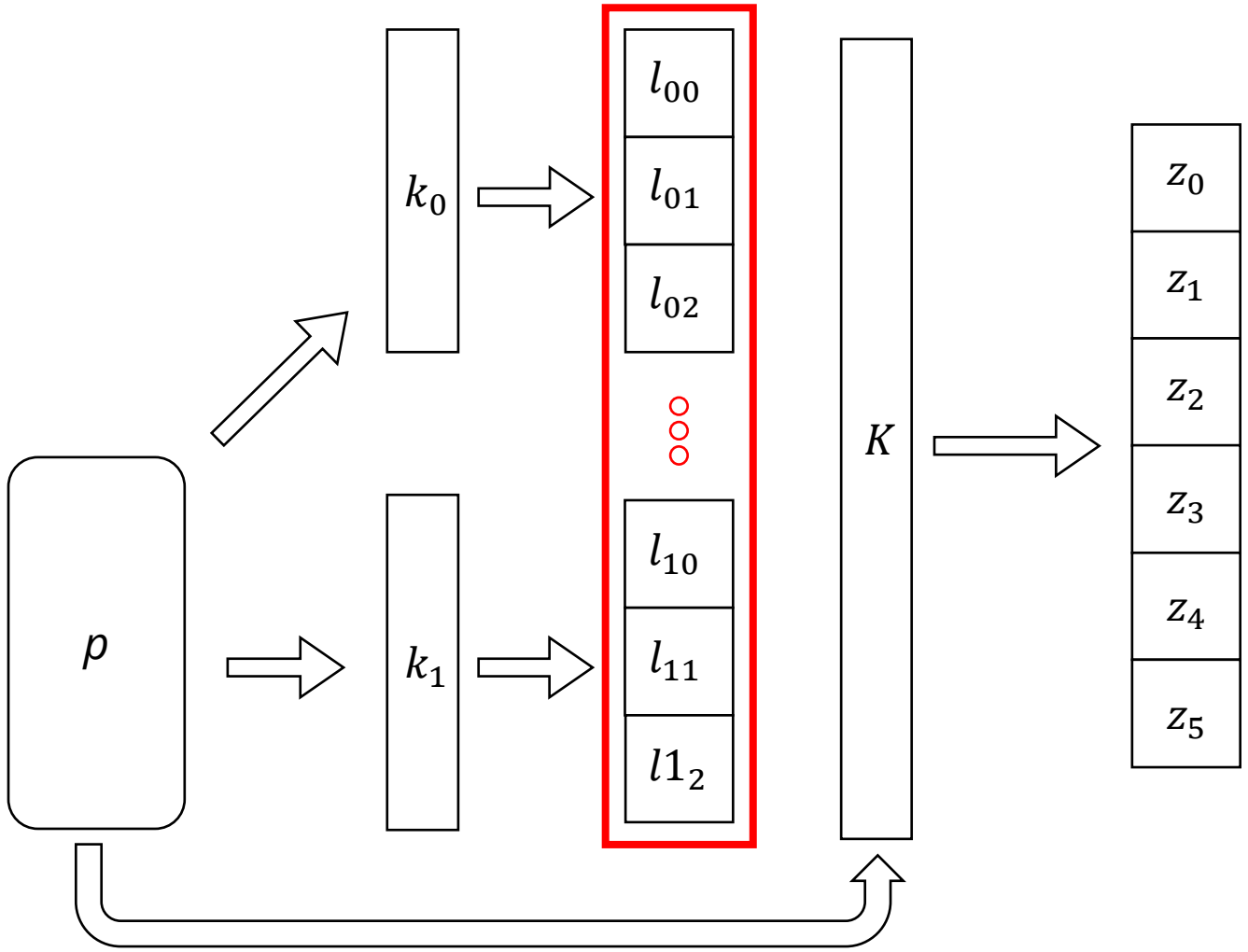
- **Модель –** на площині генеруються n прямих та m точок. Нейронна мережа має навчитись визначати, де знаходиться точка по відношенню до даної прямої (справа – 1, зліва – 0, на прямій – 2). На вихід подається вектор ознак у вигляді числа, що є конкатенацією відповідних цифр (результат відповідає класу).
- **Причини створення –** необмежена кількість прикладів для навчання та тестів, висока швидкість навчання та аналізу, можливість відстеження аномалій та невідомих класів.
- Формат навчального прикладу – [*x*; *y*; *fv*].

АДАПТИВНИЙ МЕТОД НАВЧАННЯ НА ОСНОВІ ПАРАЛЕЛЬНОЇ КОМПОЗИЦІЙНОЇ МОДЕЛІ

- Вихідна нейронна мережа складається з *k* підмереж.
- Жодна з підмереж не зв’язана з іншою синаптичними вагами.
- Кожна підмережа отримує на вхід приклад *p*.
- Результат класифікації вихідної мережі представляє собою конкатенацію векторів ознак кожної з підмереж при класифікації прикладу *p*.

ПАРАЛЕЛЬНА МОДЕЛЬ (ПРИКЛАД)

- n напередвизначених прямих, де $n = 5$;
- p – приклад, представлений у вигляді координат точки (x, y);
- K – одинична мережа;
- k_0 – підмережа, що розпізнає p відносно прямих $\{0, 1, 2\}$;
- k_1 – підмережа, що розпізнає p відносно прямих $\{3, 4, 5\}$;
- l_{ij} – j-та компонента вектора ознак, отриманого внаслідок класифікації прикладу p мережею k_i;
- z – вектор ознак, отриманий внаслідок класифікації прикладу p мережею K.

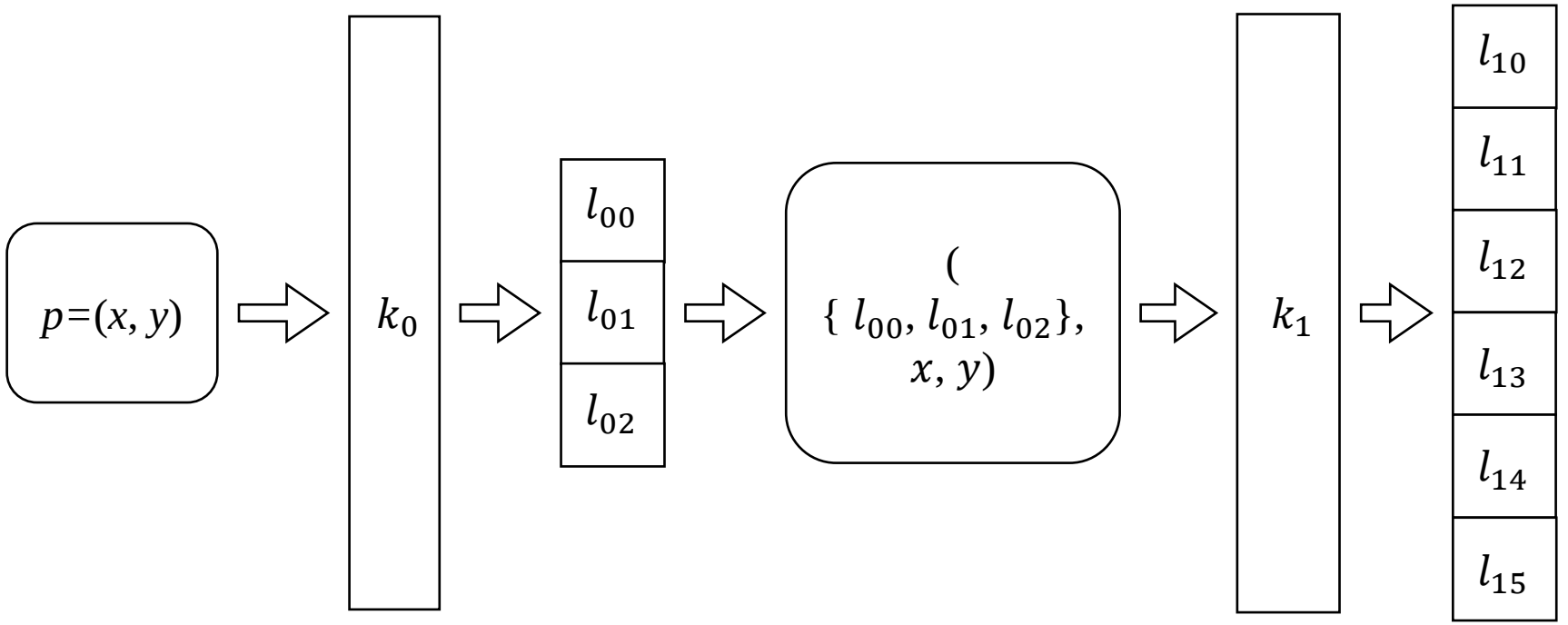


АДАПТИВНИЙ МЕТОД НАВЧАННЯ НА ОСНОВІ ПОСЛІДОВНОЇ КОМПОЗИЦІЙНОЇ МОДЕЛІ

- Вихідна нейронна мережа складається з *k* підмереж.
- Жодна з підмереж не зв’язана з іншою синаптичними вагами.
- Кожна підмережа отримує на вхід приклад *p* та вектор ознак, наданий попередньою мережею.
- Кожна наступна підмережа виконує докласифікацію ознак, за яку вона відповідає.
- Результат класифікації вихідної мережі є результатом класифікації останньої у послідовності мережі.

ПОСЛІДОВНА МОДЕЛЬ (ПРИКЛАД)

- n напередвизначених прямих, де $n = 5$;
- p – приклад, представлений у вигляді координат точки (x, y);
- $k_0 -$ підмережа, що розпізнає p відносно прямих $\{0, 1, 2\}$;
- $k_1 -$ підмережа, що розпізнає p відносно прямих $\{3, 4, 5\}$;
- $l_{ij} - j$-та компонента вектора ознак, отриманого внаслідок класифікації прикладу p мережею k_i.



*** - без використання регуляризаційних технік**

РЕЗУЛЬТАТИ НАВЧАННЯ*

Одиночна	Паралельна композиція	Послідовна композиція
16-8_50_4_0.25		
0.74	0.81 (+0.07)	0.81 (+0.07)
16-8_50_4_0.75		
0.73	0.76 (+0.03)	0.76 (+0.03)
8-16-8_50_4_0.25		
0.63	0.81 (+0.18)	0.8 (+0.17)
8-16-8_50_4_0.75		
0.53	0.77 (+0.24)	0.76 (+0.23)
8-16-8_150_8_0.25		
0.99	0.99	0.99

*** - без використання регуляризаційних технік**

РЕЗУЛЬТАТИ НАВЧАННЯ* (ПРОДОВЖЕННЯ)

Одиночна	Паралельна композиція	Послідовна композиція
8-16-8_150_8_0.75		
0.78	0.99 (+0.11)	0.99 (+0.11)
8-16-32-32-16-8_150_4_0.25		
0.66	0.81 (+0.15)	0.79 (+0.13)
8-16-32-32-16-8_150_4_0.75		
0.70	0.75 (+0.05)	0.77 (+0.07)
8-16-32-32-16-8_150_8_0.25		
0.99	0.99	0.99
8-16-32-32-16-8_150_8_0.75		
0.60	0.99 (+0.33)	0.99 (+0.33)

КІЛЬКІСТЬ ОБЧИСЛЮВАЛЬНИХ ОПЕРАЦІЙ

Структура прихованих шарів	Кількість операцій одиночної мережі	Кількість операцій композиції
16-8	128	32
8-16-8	256	64
8-16-32-32-16-8	2304	576

*** розмір міні-батчу - 128**

ПІКОВЕ СПОЖИВАННЯ ПАМ’ЯТІ*

Структура прихованих шарів	Споживання одиночної мережі	Споживання композиції
16-8	2 Гб	1 Гб
8-16-8	2.9 Гб	1.4 Гб
8-16-32-32-16-8	≈4 Гб (відмова системи)	2.1 Гб

ФОРМУЛЮВАННЯ КРИТЕРІЮ

- Сформулюємо функцію похибки наступним чином:

$$f(I, N) = \varepsilon,$$

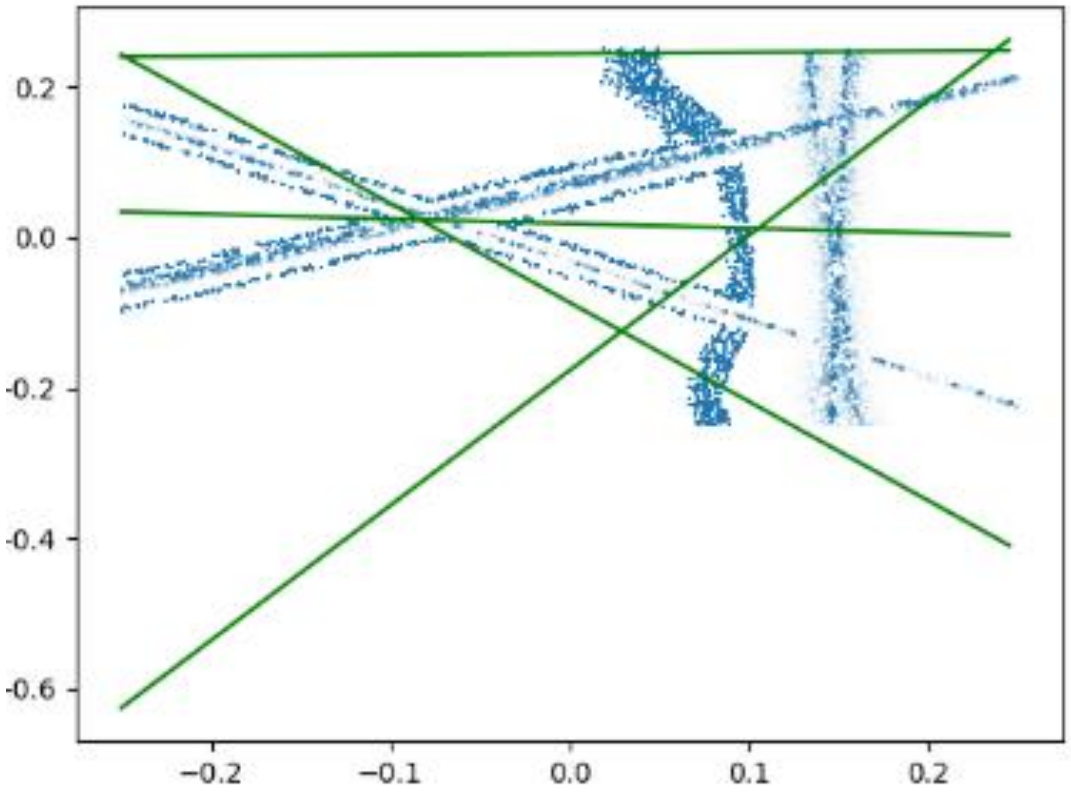
де f – функція похибки, I – набір вихідни даних, N – нейромережа, ε – середнє значення похибки на всій множині $G(\mathbf{X})$.

- Граничне значення похибки, що вибирається для кожної з задач класифікації є постійним на всій області визначення функції $G(\mathbf{X})$.

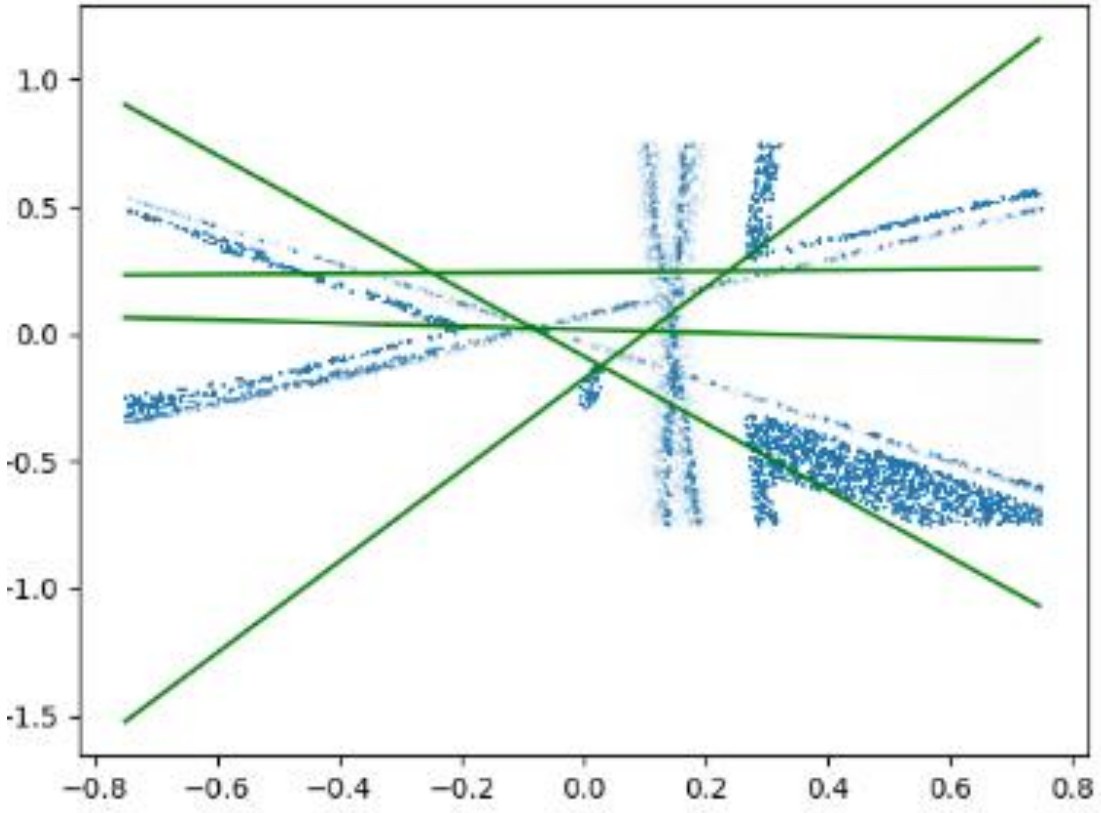
ФОРМУЛЮВАННЯ КРИТЕРІЮ (ПРОДОВЖЕННЯ)

- Результат використання критерію – поставлення у відповідність мережі одного з наступних статусів, кожний з яких обумовлює ступінь навченості:
  - Ненавчена.
  - Недонавчена.
  - Навчена.
  - Перенавчена.

РЕЗУЛЬТАТИ ЗАСТОСУВАННЯ КРИТЕРІЮ

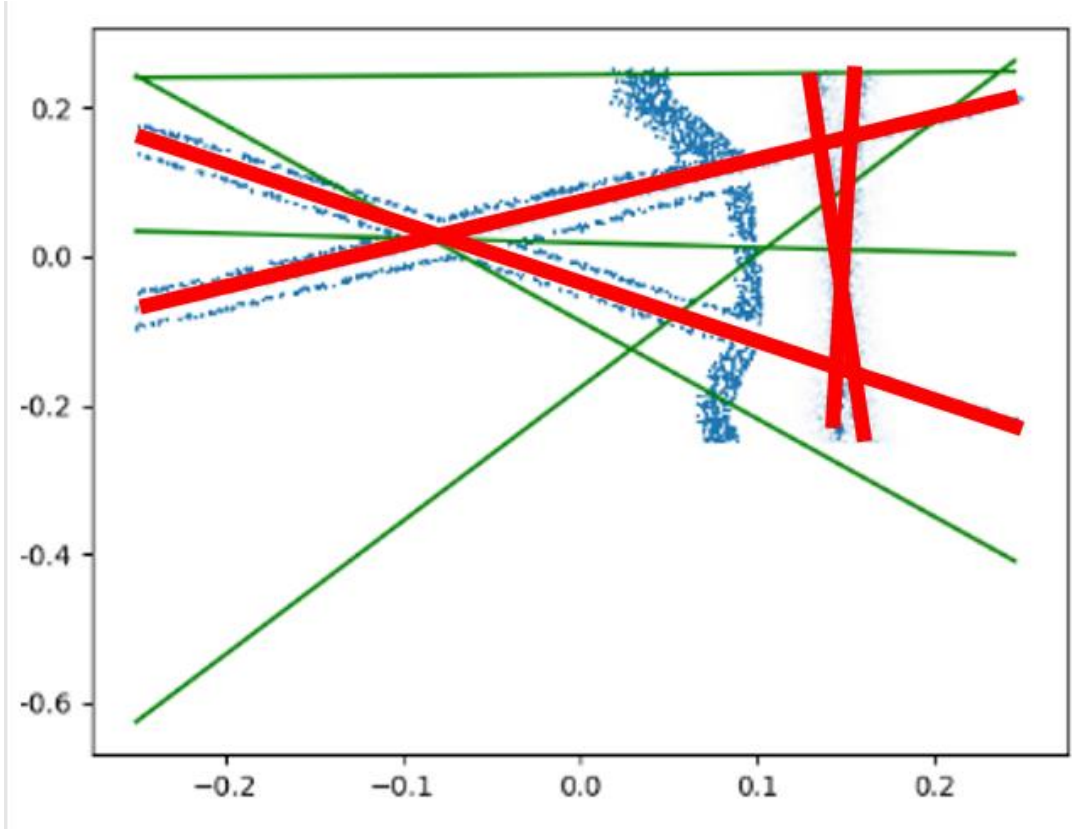


8-16-8_150_8_0.25

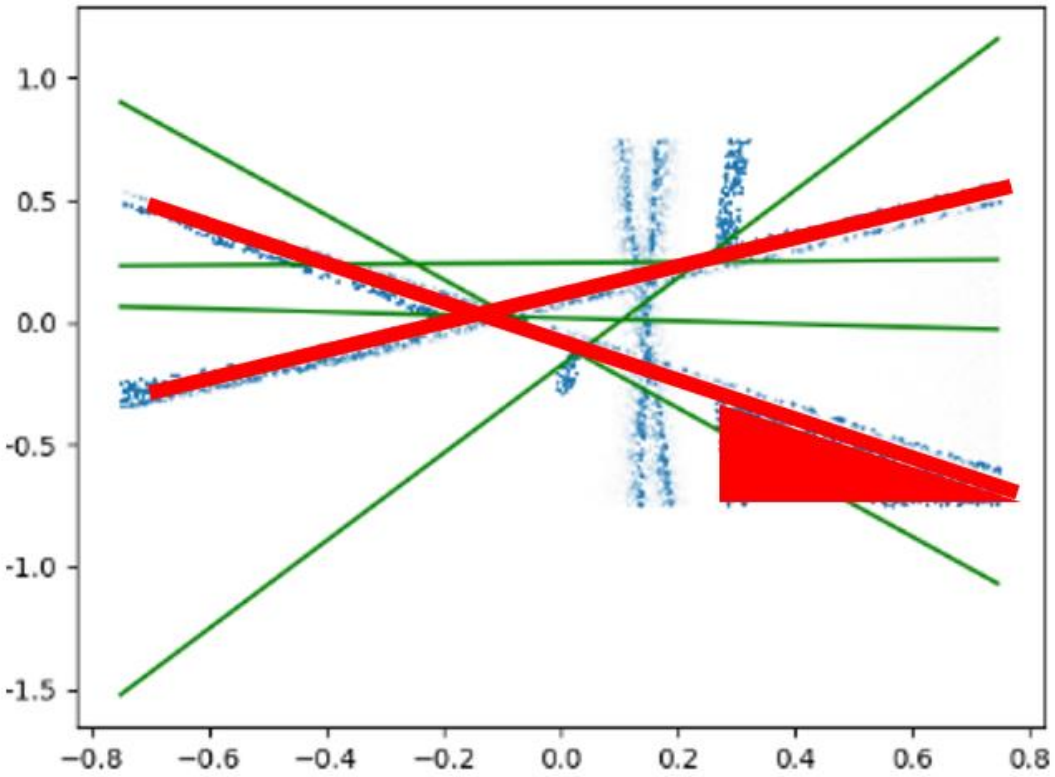


8-16-8_50_8_0.75

РЕЗУЛЬТАТИ ЗАСТОСУВАННЯ КРИТЕРІЮ (ПРОДОВЖЕННЯ)



8-16-8_150_8_0.25



8-16-8_50_8_0.75

ЕЛЕМЕНТИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

ЗАГАЛЬНА ІНФОРМАЦІЯ

- Програмне забезпечення представляє собою об’єктно-орієнтовану програмну реалізацію та включає в себе наступні модулі:
  - Модуль генерації наборів даних (реалізація модельної задачі).
  - Модуль передобробки наборів даних.
  - Модуль генерації конфігураційних файлів.
  - Модуль навчання моделей.
  - Модуль візуалізації та збереження результатів.

ВИКОРИСТАНІ МОВИ ТА БІБЛІОТЕКИ

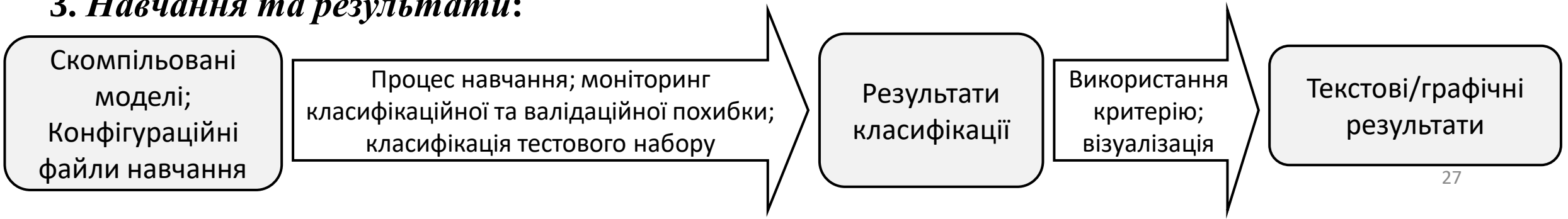
Модуль	Мови та бібліотеки
Модуль генерації наборів даних	Maxima v. 5.44, LISP
Модуль передобробки наборів даних	Python 3.7
Модуль генерації конфігураційних файлів	
Модуль навчання моделей	Python 3.7: Tensorflow & Tensorboard 2.0, Keras 2.0, numpy, Scikit-Learn
Модуль візуалізації та збереження результатів	Python 3.7: matplotlib, numpy.

СХЕМА РОБОТИ ПЗ

1. ***Генерація наборів даних*:**

2. ***Побудова моделей*:**

3. ***Навчання та результати*:**



ФОРМАТ ВХІДНИХ/ВИХІДНИХ ДАНИХ

- Кожний приклад описано як (*x*, *y*, *l*), де: *x*, *y* – координати точки, *l* – відповідний вектор ознак.

```
-0.172161754879505 8;     0.0689288930588418 7;     "1100"
-0.241664187403795 9;     0.358811817376307 5;      "1010"
...
```

- Результат класифікації описано як ([*x*, *y*]; *fv_p*; *fv*; *e*; *t*), де: *x*, *y* – координати точки, *fv_p* – результат класифікації, *fv* – вихідний вектор ознак, *e* – значення похибки, *t* – тип використовуваної похибки.

```
[-0.02048 -0.19832];[1. 1. 0. 1.];[1 1 0 1];0.0;mse
[0.12817 0.22502];[0.456437 0. 1. 0.99931043];[1 0 1 1];0.073865304875686 1;mse
...
```

НАУКОВА НОВИЗНА

- Запропоновано методи адаптивного навчання нейронних мереж за наявності обмежених обчислювальних ресурсів, засновані на використанні композиційних архітектурщо надають навченим мережам властивість адаптивності, підвищуючи точність класифікації на 10-15%.
- Запропоновано критерій оцінки навченості нейромереж, за яким можливо оцінити похибку класифікації як контінуальну множину, що дозволяє ідентифікувати піднабори прикладів, точність класифікації яких є нижчою за допустиму.
- Розроблено програмне забезпечення, що реалізовує навчання нейромереж, використовуючи методи адаптивного навчання та зазначений критерій.

ВИСНОВКИ

- Було проведено аналіз існуючих методів навчання та верифікації нейронних мереж в рамках обмежених обчислювальних ресурсів та сформовані задачі дослідження.
- Запропоновано методи адаптивного навчання нейронних мереж та критерій навченості, які було реалізовано та протестовано у вигляді програмного забезпечення.
- В рамках розробленого ПЗ була реалізована модель генерації синтетичних даних.

ВИСНОВКИ (ПРОДОВЖЕННЯ)

- Використовуючи розроблене програмне забезпечення:
  - Запропоновані методи було протестовано на різних конфігураціях нейромереж та гіперпараметрів.
  - Було виконано порівняння результатів роботи запропонованих методів із існуючими.
  - Було наведено аналіз отриманих результатів та сформовані подальші цілі дослідження.
- Методи було оцінено наступними метриками:
  - Дискретна точність класифікації навченої нейронної мережі.
  - Кількість виконаних операцій для обраних мереж.
  - Відповідність розробленого програмного забезпечення вимогам.

ПОДАЛЬША РОБОТА

- Використання методу для масштабніших структур нейронних мереж.
- Використання методу для інших видів нейронних мереж.
- Адаптація модифікованого методу до регуляризаційних технік.
- Формування принципів композиції нейромереж.
- Використання напрацьованої бази знань для розробки методів декомпозиції нейронних мереж.
- Дослідження залежність форми кластерів від форми обмежень та функції активації (як лінійної, так і нелінійної).
- Дослідження локалізації кластерів від локалізації обмежень.

АПРОБАЦІЯ РОБОТИ

- XII наукова конференція магістрантів та аспірантів «Прикладна математика та комп’ютинг» ПМК-2020 (Київ, 18-20 листопада 2020 р.).
- Стаття буде опублікована у журналі International Journal of Modern Education and Computer Science та проіндексована у Scopus.
- Очікується підтвердження на проведення відповідної конференції в червні 2021.

ДЯКУЮ ЗА УВАГУ!